DOCUMENT RESUME

ED 288 490 IR 012 881

| | |
|---|---|
| AUTHOR | Falkenhainer, Brian; And Others |
| TITLE | The Structure-Mapping Engine: Algorithm and Examples. |
| INSTITUTION | Illinois Univ., Urbana. Dept. of Computer Science. |
| SPONS AGENCY | Office of Naval Research, Arlington, Va. Personnel and Training Research Programs Office. |
| REPORT NO | UILU-ENG-87-1748; UIUCDCS-R-87-1361 |
| PUB DATE | Jul 87 |
| CONTRACT | N00014-85-K-0559 |
| NOTE | 69p.; For related document, see ED 277 544. |
| PUB TYPE | Reports - Research/Technical (143) |
| EDRS PRICE | MF01/PC03 Plus Postage. |
| DESCRIPTORS | *Algorithms; *Artificial Intelligence; Cognitive Processes; *Cognitive Structures; *Computer Software; Epistemology; *Learning Theories; Logical Thinking; Models; Sciences; Simulation |
| IDENTIFIERS | *Analogies; *Structure Mapping Engine |

ABSTRACT

This description of the Structure-Mapping Engine (SME), a flexible, cognitive simulation program for studying analogical processing which is based on Gentner's Structure-Mapping theory of analogy, points out that the SME provides a "tool kit" for constructing matching algorithms consistent with this theory. This report provides: (1) a review of the Structure-Mapping Theory; (2) a description of the design of the engine, including representation conventions, an overview of the computations performed by the algorithm, and analyses of these computations; (3) examples of the application of the SME to various analogies, including the Rutherford Atom Analogy, the PHINEAS system for discovering heat flow, and in modelling human analogical processing; (4) a comparison of the SME with other work in artificial intelligence and psychology; and (5) a discussion of the implications of the project and plans for future work in cognitive simulation and machine learning studies. Sixty-six references are listed and two appendices provide additional explanations of the three types of rule sets used by the SME and sample domain descriptions given to the SME for the described examples. (RP)

**********
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
**********

REPORT NO. UIUCDCS-R-87-1361 UILU-ENG-87-1748

# THE STRUCTURE-MAPPING ENGINE: ALGORITHM AND EXAMPLES

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

This document has been reproduced as received from the person or organization originating it.

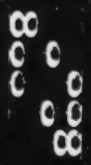
☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

by

Brian Falkenhainer
Kenneth D. Forbus
Dedre Gentner

July 1987



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

SECURITY CLASSIFICATION OF THIS PAGE

# REPORT DOCUMENTATION PAGE

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION<br>Unclassified | 1b. RESTRICTIVE MARKINGS |
| 2a. SECURITY CLASSIFICATION AUTHORITY | 3. DISTRIBUTION / AVAILABILITY OF REPORT<br>Approved for public release;<br>Distribution unlimited |
| 2b. DECLASSIFICATION / DOWNGRADING SCHEDULE | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S)<br>UIUCDCS-R-87-1361 | 5. MONITORING ORGANIZATION REPORT NUMBER(S) |
| 6a. NAME OF PERFORMING ORGANIZATION<br>University of Illinois<br>Dept. of Computer Science<br>6b. OFFICE SYMBOL (If applicable) | 7a. NAME OF MONITORING ORGANIZATION<br>Personnel and Training Research Programs<br>Office of Naval Research (Code 1142PT) |
| 6c. ADDRESS (City, State, and ZIP Code)<br>1304 W. Springfield<br>Urbana, Illinois 61801 | 7b. ADDRESS (City, State, and ZIP Code)<br>800 N. Quincy Street<br>Arlington, Virginia 22217-5000 |
| 8a. NAME OF FUNDING / SPONSORING ORGANIZATION<br>8b. OFFICE SYMBOL (If applicable) | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER<br>N00014-85-K-0559 |
| 8c. ADDRESS (City, State, and ZIP Code) | 10. SOURCE OF FUNDING NUMBERS |

| PROGRAM ELEMENT NO. | PROJECT NO. | TASK NO. | WORK UNIT ACCESSION NO |
|---|---|---|---|
| 61153N | RR04206 | RR04206-0A | NR442a551 |

11. TITLE (Include Security Classification)

The Structure-Mapping Engine: Algorithm and Examples

12. PERSONAL AUTHOR(S)

Brian Falkenhainer, Kenneth D. Forbus, Dedre Gentner

| 13a. TYPE OF REPORT | 13b. TIME COVERED | 14. DATE OF REPORT (Year, Month, Day) | 15. PAGE COUNT |
|---|---|---|---|
| Technical | FROM 85-9-1 TO 88-8-31 | July 1987 | 59 |

16. SUPPLEMENTARY NOTATION

| 17. COSATI CODES | | |
|---|---|---|
| FIELD | GROUP | SUB-GROUP |
| 05 | 08 | |
| | | |

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

analogy
Artificial Intelligence
machine learning
matching

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This paper describes the Structure-Mapping Engine (SME), a program for studying analogical processing. SME is based on Gentner's Structure-Mapping theory of analogy, and provides a "tool kit" for constructing matching algorithms consistent with this theory. Its flexibilit enhances cognitive simulation studies by simplifying experimentation. Furthermore, SME is very efficient, making it a useful component in machine learning systems as well. We review the Structure-Mapping theory and describe the design of the engine. We analyze the complexity of the algorithm, and demonstrate that most of the steps are polynomial, typically bounded by $O(N^2)$. Next we demonstrate some examples of its operation taken from our cognitive simulation studies and work in machine learning. Finally, we compare SME to other analogy programs and discuss several areas for future work.

| 20. DISTRIBUTION / AVAILABILITY OF ABSTRACT | 21. ABSTRACT SECURITY CLASSIFICATION |
|---|---|
| ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS | Unclassified |

| 22a. NAME OF RESPONSIBLE INDIVIDUAL | 22b. TELEPHONE (Include Area Code) | 22c. OFFICE SYMBOL |
|---|---|---|
| Dr. Susan Chipman | (202) 696-4318 | ONR 1142PT |

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

# THE STRUCTURE-MAPPING ENGINE: ALGORITHM AND EXAMPLES

Brian Falkenhainer
Kenneth D. Forbus
Dedre Gentner

July 1987

Department of Computer Science
University of Illinois at Urbana-Champaign
1304 W. Springfield Avenue
Urbana, Illinois 61801

This research is supported by the Office of Naval Research, Personnel and Training Research Programs, Contract No. N00014-85-K-0559.

Approved for public release; distribution unlimited.

Submitted for publication.

## Contents


**1 Introduction** 1

**2 Structure-Mapping Theory** 2
- 2.1 Constraints on Analogy . . . 2
- 2.2 Other types of similarity . . . 4
- 2.3 Subprocesses in analogy . . . 5
- 2.4 Empirical evidence . . . 6

**3 The Structure-Mapping Engine** 6
- 3.1 Representation conventions . . . 7
  - 3.1.1 Entities . . . 7
  - 3.1.2 Predicates . . . 7
  - 3.1.3 Facts and Dgroups . . . 9
- 3.2 The SME Algorithm: Overview . . . 10
  - 3.2.1 Step 1: Local match construction . . . 11
  - 3.2.2 Step 2: Global Match Construction . . . 14
  - 3.2.3 Step 3: Compute Candidate Inferences . . . 18
  - 3.2.4 Step 4: Compute Structural Evaluation Scores . . . 19
- 3.3 Analysis . . . 23
  - 3.3.1 Analysis of local match construction . . . 24
  - 3.3.2 Analysis of *Conflicting* calculation . . . 24
  - 3.3.3 Analysis of *EMaps* and *NoGood* calculation . . . 25
  - 3.3.4 Analysis of Gmap merge steps . . . 25
  - 3.3.5 Analysis of Finding Candidate Inferences . . . 26
  - 3.3.6 Analysis of Structural Evaluation Score computation . . . 26

**4 Examples** 27
- 4.1 Methodological constraints . . . 27
- 4.2 Solar System - Rutherford Atom Analogy . . . 28
- 4.3 Discovering Heat Flow . . . 29
- 4.4 Modelling Human Analogical Processing . . . 32
- 4.5 Review of Performance . . . 35

**5 Comparison With Other Work** 35
- 5.1 Matching Algorithms . . . 37

**6 Discussion** 38
- 6.1 Implications for representation . . . 38
- 6.2 Addressing the Combinatorics . . . 39
  - 6.2.1 Medium-grained Parallel Architectures . . . 39
  - 6.2.2 Connectionist Architectures . . . 40
- 6.3 Future Work . . . 41
  - 6.3.1 Cognitive Simulation . . . 41
  - 6.3.2 Machine Learning Studies . . . 42

**7 Acknowledgements** 42

**A SME Match Rules** 47
A.1 Literal Similarity (LS) Rules . . . . . 47
A.2 Analogy (AN) Rules . . . . . 48
A.3 Mere Appearance (MA) Rules . . . . . 49

**B Sample Domain Descriptions** 50
B.1 Simple Water Flow - Heat Flow . . . . . 50
B.2 Solar-System - Rutherford Atom . . . . . 51
B.3 Karla Stories . . . . . 52

# 1 Introduction

Analogy is a computational process in which a given situation is understood by bringing to bear knowledge of previous, similar experiences. Analogy may be used to guide reasoning, to generate conjectures about an unfamiliar domain, or to generalize several experiences into an abstract schema. Consequently, analogy is of great interest to both cognitive psychologists and artificial intelligence researchers. Psychologists wish to clarify the mechanisms underlying analogy in order to understand human learning and reasoning. Artificial Intelligence researchers wish to emulate analogical processing on computers to produce more flexible reasoning and learning systems.

This paper describes the *Structure-Mapping Engine* (SME), a program built to explore the computational aspects of Gentner's *Structure-Mapping theory* of analogical processing [23,25]. SME has been used both as a cognitive simulation of human analogical processing and as a component in a larger machine learning system.

SME is both flexible and efficient. It constructs all consistent ways to interpret a potential analogy and does so without backtracking. SME provides a "tool kit" for constructing matchers consistent with the kinds of comparisons sanctioned by Gentner's theory. A matcher is specified by a collection of rules, which indicate what things might match and estimate how strongly these matches should be believed. The program uses these estimates and a novel procedure for combining the local matches constructed by the rules to efficiently produce and evaluate all consistent global matches. This efficiency and flexibility makes the matching algorithm promising for exploration of both the space of cognitive models and the computational aspects of analogy for AI.

Cognitive simulation studies can offer important insights for understanding the human mind. They serve to verify psychological theories and force one to pin down those aspects of a theory which might otherwise be left unspecified. They also offer unique opportunities to construct idealized subjects, whose prior knowledge and set of available processes is completely known to the experimenter. Unfortunately, cognitive simulation programs tend to be complex and computationally expensive (c.f. [1,59]). Complexity can obscure the relationship between the theory and the program. Typically there are many design decisions in building a program, and if one cannot assign credit to them when analyzing results then it can be hard to see where the performance is really coming from. Often it is desirable to explore a space of similar architectures to determine what the consequences of particular design decisions are and to model particular performance in detail. Such explorations are very difficult if the major way to change the program's operation is surgery on the code. Being computationally expensive means performing fewer experiments, and thus exploring fewer possibilities. While there have been several important AI programs that study computational aspects of analogy (e.g., [3,65,66]), they were not designed to satisfy the above criteria.

Recently there has been a plethora of approaches to analogy in AI (as we review later), but surprisingly little progress so far. Often papers describe programs that work on only a handful of carefully chosen examples, and do not specify the algorithms in a replicable fashion. We believe the reason so little progress has been made is that analogy is a complex problem, and that the appropriate decomposition is critical. Without a good decomposition, it is easy to tackle several semi-independent problems at once, or an underconstrained aspect of the problem, and become lost in the space of possible mechanisms. Our decomposition, described in the next section, is psychologically motivated. Roughly, SME focuses on the *mapping* aspect of analogy, leaving the *access* and *application* aspects to future studies. The power of the program that results, and its success on a wide variety of examples (over 40 as of this writing), provides additional evidence that

the decomposition is a good one.

This paper examines the architecture of the Structure-Mapping Engine and how it has been used for machine learning and cognitive simulation. First, we review Gentner's Structure-Mapping theory and some of the psychological evidence for it. Next we discuss the organization of SME, including the knowledge representation conventions and the details of the algorithm. After a complexity analysis, we then illustrate SME's operation on several examples drawn from machine learning and cognitive simulation studies. Related work in both AI and psychology is reviewed next, followed by a discussion of further issues raised by this design.

## 2 Structure-Mapping Theory

The theoretical framework for this research is Gentner's Structure-Mapping theory of analogy [23,24,25,26,27,28]. Structure-Mapping describes the set of implicit rules by which people interpret analogy and similarity. The central idea is that an analogy is a mapping of knowledge from one domain (the base) into another (the target) which conveys that a system of relations known to hold in the base also holds in the target. The target objects do not have to resemble their corresponding base objects. Objects are placed in correspondence by virtue of corresponding roles in the common relational structure.

This structural view of analogy is based on the intuition that analogies are about relations, rather than simple features. No matter what kind of knowledge (causal models, plans, stories, etc.), it is the structural properties (i.e., the interrelationships between the facts) that determine the content of an analogy. For example, consider the heat flow and water flow situations shown in Figure 1. These situations are thought to be analogous because they share the complex relationship known as "flow". In each, we have a rough picture of something flowing downhill, from a source to a destination. We prefer to ignore the appearances and even specific defining properties of the objects, such as the fact that water and coffee are both liquids. Indeed, focusing on these attributes tends to confuse our picture of the analogy.

### 2.1 Constraints on Analogy

An important preliminary: We define the *order* of an item in a representation as follows: Objects and constants are order 0. The order of a predicate is one plus the maximum of the order of its arguments. Thus `GREATER-THAN(x,y)` is first-order if x and y are objects, and `CAUSE[GREATER-THAN(x,y), BREAK(x)]` is second-order. Examples of higher-order relations include `CAUSE` and `IMPLIES`. This definition of order should *not* be confused with the standard definition.[1] Essentially, we use this definition of order to indicate how deep the structure is below an item. Notice that intricate arguments with many layers of justifications will give rise to representation structures of high order.

Given collections of objects $\{b_i\}$, $\{t_i\}$ in the base and target representations, respectively, the tacit constraints on the analogical mapping M can be characterized as follows:

1. Objects in the base are placed in correspondence with objects in the target:

   M: $b_i \rightarrow t_i$

[1] Under the standard definition, a logic is first-order if variables only range over objects and second-order when it permits variables to range over predicates as well.

Figure 1: Two physical situations involving flow.

2. Isolated object descriptions are discarded unless they are involved in a larger relational structure.

   e.g. $\texttt{RED}(b_i) \not\longrightarrow \texttt{RED}(t_i)$

3. Relations between objects in the base tend to be mapped across:

   e.g. $\texttt{COLLIDE}(b_i, b_j) \longrightarrow \texttt{COLLIDE}(t_i, t_j)$

4. The particular relations mapped are determined by *systematicity*, as defined by the existence of higher-order constraining relations which can themselves be mapped:

   e.g. $\texttt{CAUSE}[\texttt{PUSH}(b_i, b_j), \texttt{COLLIDE}(b_j, b_k)] \Longrightarrow$
   $\texttt{CAUSE}[\texttt{PUSH}(t_i, t_j), \texttt{COLLIDE}(t_j, t_k)]$

For example, consider the analogy between heat-flow and water-flow. Figure 2 shows what a learner might know about the domains pictured in Figure 1. In order to comprehend the analogy "Heat is like water" a learner must do the following (although not necessarily in this order):

1. Set up the object correspondences between the two domains:

   `heat → water, pipe → metal bar, beaker → coffee, vial → ice cube`

2. Discard object attributes, such as `LIQUID(water)`.

3. Map base relations such as

   `GREATER-THAN[PRESSURE(beaker), PRESSURE(vial)]`

   to the corresponding relations in the target domain.

4. Observe systematicity: i.e., keep relations belonging to a systematic relational structure in preference to isolated relationships. In this example,

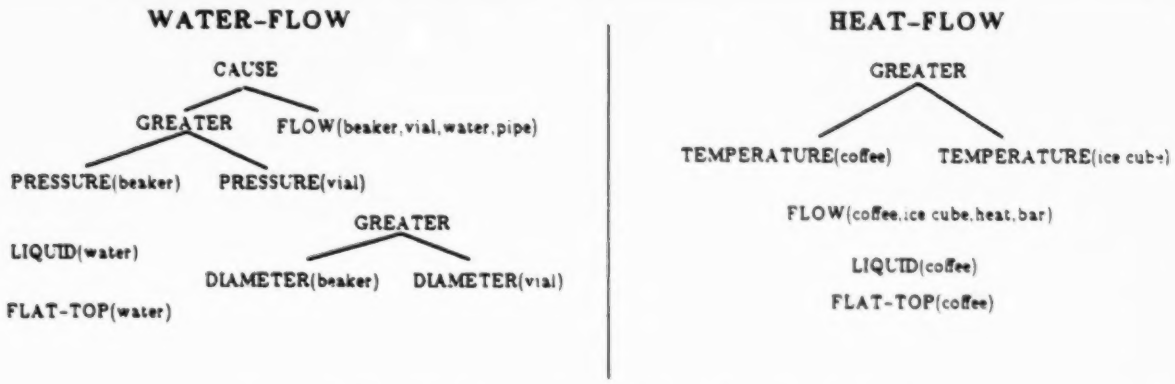



**Figure 2: Simplified water flow and heat flow descriptions.**

```
CAUSE(GREATER-THAN[PRESSURE(beaker),
                   PRESSURE(vial)],
      FLOW(beaker, vial, water, pipe))
```

is mapped into

```
CAUSE(GREATER-THAN[TEMPERATURE(coffee),
                   TEMPERATURE(ice cube)],
      FLOW(coffee, ice cube, heat, bar))
```

while isolated relations, such as

```
GREATER-THAN[DIAMETER(beaker), DIAMETER(vial)]
```

are discarded.

The *systematicity* principle is central to analogy. Analogy conveys a system of connected knowledge, not a mere assortment of independent facts. Preferring systems of predicates that contain higher-order relations with inferential import is a structural expression of this tacit preference for coherence and deductive power in analogy. Thus, it is the amount of common, higher-order relational structure that determines which of several possible matches is preferred. For example, suppose in the previous example we were concerned with objects differing in specific heat, such as a metal ball-bearing and a marble of equal mass, rather than temperatures. Then DIAMETER would enter the mapping instead of (or in addition to) PRESSURE, since DIAMETER affects the capacity of a container, the analog to specific heat.

## 2.2 Other types of similarity

In addition to analogy, Structure-Mapping theory defines several other kinds of similarity. As we have seen, in *analogy* only higher-order relations are mapped. Aspects of object descriptions which

play no role in the relational structure are ignored. By contrast, in *literal similarity* both relational predicates and object-descriptions are mapped. Literal similarity typically occurs in within-domain comparisons, where the objects involved look alike as well as act alike. An example of a literal similarity is the comparison "Kool-Aid is like water."[2] In *mere-appearance* matches, it is primarily the object-descriptions which are mapped, as in the metaphor

"Her eyes were like the deep blue of a summer sky."

A fourth kind of mapping is the *abstraction mapping*. Here, the entities in the base domain are variables, rather than objects. Few, if any, attributes exist that do not contribute to the base's relational structure. The result of an abstraction matching is very close to the instantiation of a rule. The difference is that only entities may be variables, whereas in many pattern-directed rule systems predicates may be used in substitutions as well.

It should be clear that Structure-Mapping neither subsumes unification, nor is subsumed by it. For example, the pair of statements

```
(CAUSE (FLY PERSON1) (FALL PERSON1))
(CAUSE (FLY PERSON2) (FALL PERSON2))
```

could be part of an analogy, with PERSON1 being mapped to PERSON2, but these two statements do not unify since PERSON1 and PERSON2 are distinct constants. Conversely,

```
(CAUSE (?X PERSON1) (FALL PERSON1))
(CAUSE (FLY ?Y) (FALL ?Z))
```

will unify, assuming ? indicates variables, with the substitutions:

```
?X ↔ FLY
?Y ↔ PERSON1
?Z ↔ PERSON1
```

However, since Structure-Mapping treats variables as constants, these statments fail to be analogous in two ways. First, FLY and ?X are treated as distinct relations, and thus cannot match. Second, ?Y and ?Z are considered to be distinct entities, and thus are forbidden to map to the same target item (i.e., PERSON1).

## 2.3 Subprocesses in analogy

Structure-Mapping decomposes analogical processing into three basic stages ([29,22,26]):

1. *Access*: Given a current *target* situation, one must first retrieve from long-term memory knowledge of another description, the *base*, which is analogous or similar to the *target*.
2. *Mapping*: This stage establishes correspondences between the base and target. Potentially, there is additional knowledge in the base that can be transferred to the target. This additional knowledge is the set of *candidate inferences* sanctioned by the analogy.

[2]Notice that our characterisation of literal similarity is still structural, and thus is differs from other psychological approaches (e.g., [56]).

3. ***Evaluation and Use***: There are two kinds of criteria for evaluating the quality of a match. The structural criteria include the number of similarities and differences, the degree of structural similarity involved, and the amount and type of new knowledge or insight the analogy provides via the candidate inferences. The second kind of criteria concerns the validity of the match and the inferences it sanctions. The inferences must be checked against current world knowledge to ensure that the analogy at least makes sense, and may require additional inferential work to refine the results. Detailed characterization of the validity criteria lie outside the range of the Structure-Mapping theory.

The Structure-Mapping Engine emulates the mapping stage of analogy and provides a structural, domain-independent evaluation of the match. While we believe it can be used to model access, and provides useful results for accounts of evaluation and use (see [13,14]), we will ignore these issues for most of this paper.

### 2.4 Empirical evidence

Although the focus of this paper is on computational modeling, one set of psychological findings is particularly relevant. Empirical psychological studies have borne out the prediction that systematicity is a key element of people's implicit rules for analogical mapping. Adults focus on shared systematic relational structure in interpreting analogy. They tend to include relations and omit attributes in their interpretations of analogy, and they judge analogies as more sound and more apt if base and target share systematic relational structure [23,29,30]. In developmental work, it has been found that eight-year olds (but not five-year olds) are better at performing difficult mappings when the base structure is systematic [31].

## 3 The Structure-Mapping Engine

A simulation of Gentner's theory has been implemented in the Structure-Mapping Engine (SME). Given descriptions of a base and target, SME constructs all structurally consistent mappings between them. The mappings consist of pairwise matches between predicates and entities in the base and target, plus the set of analogical inferences sanctioned by the mapping. SME also provides a structural evaluation score for each mapping, allowing easy selection of the "best" mapping, according to the rules of systematicity and consistency.

Importantly, SME is not "hard wired", but provides a testbed for implementing matchers consistent with Gentner's Structure-Mapping theory. In addition to analogy, SME can be used to simulate the other comparisons sanctioned by the Structure-Mapping theory. Such matchers may also serve a valuable role in reasoning or learning programs. For example, given the descriptions of water flow and heat flow shown in Figure 2, SME would offer several alternative interpretations for this potential analogy. In one interpretation, the central inference is that water flowing from the beaker to the vial corresponds to heat flowing from the coffee to the ice cube. Alternatively, one could map water to coffee, since they are both liquids. This is an interpretation provided by SME, but with a lower evaluation score. A learning system could use the structural evaluation to select the explanation most likely intended by the human teacher.

This section describes the SME algorithm in sufficient detail to allow replication. We start by defining some simple conventions for knowledge representation, since these conventions are essential to understanding the algorithm.

## 3.1 Representation conventions

We make as few representational assumptions as possible so that SME remains domain-independent. However, a few conventions are necessary. We use a typed (higher-order, in the standard sense) predicate calculus to represent facts. The constructs of this language are:

**Entities:** Individuals and constants.

**Predicates:** There are three types: *functions*, *attributes*, and *relations*. Each is described below.

**Dgroup:** A *description group* is a collection of entities and facts about them, considered as a unit.

We examine each construct in turn.

### 3.1.1 Entities

Entities are logical individuals, i.e., the objects and constants of a domain. Typical entities include physical objects, their temperature, and the substance they are made of. Primitive entities are declared with the `defEntity` form:

```
(defEntity ⟨name⟩
   [:type ⟨EntityType⟩]
   [:constant?  {t | nil}] )
```

Since our language is typed, each entity type can be declared as a subtype of an existing type using the `:type` option. For example, we might have

```
(defEntity star :type inanimate)
(defEntity Sun :type star)
```

to say that stars are inanimate objects, and our Sun is a particular star. Constants are declared by using the `:constant?` option, as in

```
(defEntity zero :type number :constant? t)
```

### 3.1.2 Predicates

We first describe the three types of predicates, and then show how they are declared.

**Functions** Functions map one or more entities into another entity or constant. For example, `(PRESSURE piston)` maps the physical object `piston` into the quantity which describes its pressure. We treat functions whose range are truth values as relations (see below), rather than functions. Consequently, Structure-Mapping treats functions differently from other types of predicates. It allows substitution of functions to acknowledge their role as an indirect way of referring to entities.

**Attributes** An attribute describes some property of an entity. Examples of attributes include `RED` and `CIRCLE`. It is well-known that a combination of a function and a constant is logically equivalent to an attribute. For example,

```
(RED BlockA)
(= (COLOR BlockA) RED)
```

and

```
(SQUARE BlockA)
(= (SHAPE BlockA) SQUARE)
```

are logically equivalent. However, these two forms do not behave identically under Structure-Mapping. We assume that a reasoner has a particular piece of attribute information represented in one form or another, but not both, at any particular time (We defer detailed discussion of this issue until Section 6.1).

**Relations** Like attributes, relations range over truth values. However, the arguments to relations can be other predicates as well as entities. Examples of relations include `CAUSE` and `GREATER-THAN`.

The `defPredicate` form declares predicates. It has several options, due in part to the existence of several types of predicates:

```
(defPredicate ⟨Name⟩ ⟨ArgumentDeclarations⟩ ⟨PredicateType⟩
   :expression-type ⟨DefinedType⟩
   [:commutative? {t | nil}]
   [:lexpr? {t | nil}] )
```

⟨*PredicateType*⟩ is either `function`, `attribute`, or `relation`, according to what kind of predicate ⟨*Name*⟩ is. The ⟨*ArgumentDeclarations*⟩ allows the arguments to be named and typed. For example, the declaration:

```
(defPredicate CAUSE ((antecedent event) (consequent event)) relation)
```

states that `CAUSE` is a two-place relational predicate. Its arguments are called `antecedent` and `consequent`, both of type `event`. The names and types of arguments are for the convenience of the representation builder, and are not currently used by SME. However, the type of predicate is very important to the algorithms, as we will see below.

The optional declarations `:commutative?` and `:lexpr?` provide SME with important syntactic information. `:commutative?` indicates that the predicate is commutative, and thus the order of arguments is unimportant when matching. `:lexpr?` indicates that the predicate can take any number of arguments[3]. Examples of commutative lexpr predicates include `AND`, `OR`, and `SUM`.

[3]The term derives from early lisp dialects which classified functions that took an arbitrary number of arguments as lexprs.

### 3.1.3 Facts and Dgroups

For simplicity, predicate instances are called `facts`. Notice we include terms corresponding to function applications as facts. The reason is that in discussing SME's operation we will often need to refer to them by a short name. We call the predicate of the fact its *functor*.

A *Description Group*, or *Dgroup*, is a collection of primitive entities and facts concerning them. Dgroups are defined with the `defDescription` form:

```
(defDescription ⟨DescriptionName⟩
   entities (⟨Entity1⟩, ⟨Entity2⟩,...,⟨Entityi⟩)
   facts    (⟨FactDeclarations⟩))
```

where ⟨*FactDeclarations*⟩ take the form

⟨*fact*⟩ or
(⟨*fact*⟩ `:name` ⟨*FactName*⟩)

For example, the description of water flow depicted in Figure 2 was given to SME as

```
(defDescription simple-water-flow
      entities (water beaker vial pipe)
      facts  (((flow beaker vial water pipe) :name wflow)
              ((pressure beaker) :name pressure-beaker)
              ((pressure vial) :name pressure-vial)
              ((greater pressure-beaker pressure-vial) :name >pressure)
              ((greater (diameter beaker) (diameter vial)) :name >diameter)
              ((cause >pressure wflow) :name cause-flow)
              (flat-top water)
              (liquid water)))
```

In addition, the description of heat flow depicted in Figure 2 was given to SME as

```
(defDescription simple-heat-flow
      entities (coffee ice-cube bar heat)
      facts  (((flow coffee ice-cube heat bar) :name hflow)
              ((temperature coffee) :name temp-coffee)
              ((temperature ice-cube) :name temp-ice-cube)
              ((greater temp-coffee temp-ice-cube) :name >temperature)
              (flat-top coffee)
              (liquid coffee)))
```

Notice that each fact does not need to be declared separately; SME will automatically create and name facts corresponding to `(diameter beaker)` and `(diameter vial)`.

We will refer to the facts and entities in a dgroup collectively as *items*. To describe the SME algorithm we need some terminology to express the structural relations between items. These expressions are rooted directed acyclic graphs, so we adopt standard graph-theory terminology. First, the *offspring* of a fact are its arguments. Entities have no offspring. An item $I_1$ which is in

the transitive closure (arguments of arguments, etc.) of another item $I_2$ is said to be a *descendant* of $I_2$, while $I_2$ is said to be an *ancestor* of $I_1$. An item with no ancestors is called a *root*. The term *Tree(I)* refers to the subtree starting at $I$. Notice that in Structure-Mapping the order of a predicate is inversely proportional to its depth.

## 3.2 The SME Algorithm: Overview

Given descriptions of a base and a target, represented as Dgroups, SME builds all structurally consistent interpretations of the comparison between them. Each interpretation of the match is called a *global mapping*, or *Gmap*[4]. Gmaps consist of three parts:

1. *A Global Match:* A set of pairwise matches between the facts and entities of the two Dgroups.
2. *Candidate Inferences:* A set of new facts which the comparison suggests holds in the target Dgroup.
3. A numerical *evaluation score* based on the Gmap's structural properties.

Following the Structure-Mapping theory, only purely structural criteria are used to construct and evaluate the mappings. SME has no other knowledge of either base or target domain. Neither rules of inference nor even logical connectives themselves are "wired in" to the algorithm. Each candidate inference must be interpreted as a surmise, rather than a valid conclusion. The evaluation score reflects the aesthetics of the particular type of comparison, not validity or potential usefulness. Testing the validity of candidate inferences and determining the utility of a match are left to other modules, as described in Section 2. This decomposition leads to strikingly good computational performance — better than algorithms which use additional criteria, as we argue below in Section 5.

*Match rules* specify what pairwise matches are possible and provide local measures of evidence used in computing the evaluation score. These rules are the key to SME's flexibility. To build a new matcher one simply loads a new set of match rules. This has several important advantages. First, we can simulate all of the types of comparisons sanctioned by Structure-Mapping theory with one program. Second, we could in theory "tune" the rules if needed to simulate particular kinds of human performance (although, importantly, this flexibility has not been needed so far!). Third, we can also simulate certain other analogy systems (including [35,65], as described below) for comparison purposes. In our experiments using SME, we currently use three rule sets, depending on the phenomenon being investigated. One set of rules focuses on object descriptions and is called the *mere-appearance* rules. In addition, the *analogy* rule set prefers relations, while the *literal similarity* rules look at both relations and object descriptions. Because the literal similarity rules are the most inclusive, we will be using them in this section to explain the system. Subsequent examples will then focus on *analogy* matches.

The SME algorithm is logically divided into four stages.

1. *Local match construction:* Finds all pairs of $(\langle BaseItem \rangle, \langle TargetItem \rangle)$ that potentially can match. A *Match Hypothesis* is created for each such pair to represent the possibility that this local match is part of a global match.

[4]The definition of Gmap is inspired in part by de Kleer's work on Assumption-based Truth Maintenance, although we do not use an ATMS in the actual code. The idea of combining local solutions by constructing maximally consistent sets is analogous to the process of *interpretation construction* in an ATMS. As explained below, we also find bit-vectors a useful implementation technique for carrying out set operations needed to maintain structural consistency.

2. ***Gmap construction:*** Combines the local matches into maximal consistent collections of correspondences.

3. ***Candidate inference construction:*** Derives the inferences suggested by each Gmap.

4. ***Match Evaluation:*** Attaches evidence to each local match and uses this evidence to compute structural evaluation scores for each Gmap.

We now describe each computation in detail, using a simple example to illustrate their operation.

### 3.2.1 Step 1: Local match construction

Given two dgroups, SME begins by finding potential matches between items in the base and target (see Figure 3). What can match is specified by *match constructor* rules, which take the form:

```
(MHCrule (⟨Trigger⟩ ⟨?BaseVariable⟩ ⟨?TargetVariable⟩
                    [:test ⟨TestForm⟩])
    ⟨Body⟩)
```

There are two types of constructor rules, each indicated by a different value for ⟨*Trigger*⟩. The first type of rule is indicated by a `:filter` trigger. These rules are applied to each pair of base and target facts, executing the code in ⟨*Body*⟩. If the `:test` option is used, ⟨*TestForm*⟩ must return true for the body to be run. For example, to state that a fact in the base may match a fact in the target whose functor is identical, we write:

```
(MHCrule (:filter ?b ?t :test (equal (fact-functor ?b)
                                     (fact-functor ?t)))
      (install-MH ?b ?t))
```

The second type of rule is indicated by a trigger of `:intern`. These rules are run on each match hypothesis as it is created. Typically they create match hypotheses between any functions or entities which are the arguments of the facts joined by the match hypothesis that triggered the rule.

Currently we use three sets of rules, one for analogy, one for literal similarity, and one for mere appearance matches. The current literal similarity rule set uses only three match constructor rules. One rule is the filter rule shown above. The other two are intern rules, listed in Appendix A. The content of the first intern rule is, roughly,

> "If the match hypothesis concerns two facts, then create a match hypothesis between any corresponding arguments that are both functions or entities."

The second intern rule is a specialization of the first that runs only on commutative predicates (i.e., the "*corresponding* arguments" condition is removed). The filter rule suffices to build match hypotheses between facts; the intern rules introduce the match hypotheses between entities and functions sanctioned by these fact matches.

The rules for the other types of comparisons are similar. The analogy rules only create matches between attributes when they are part of some higher-order structure. The mere appearance rule set completely ignores higher-order structure. All three rule sets are listed in Appendix A.

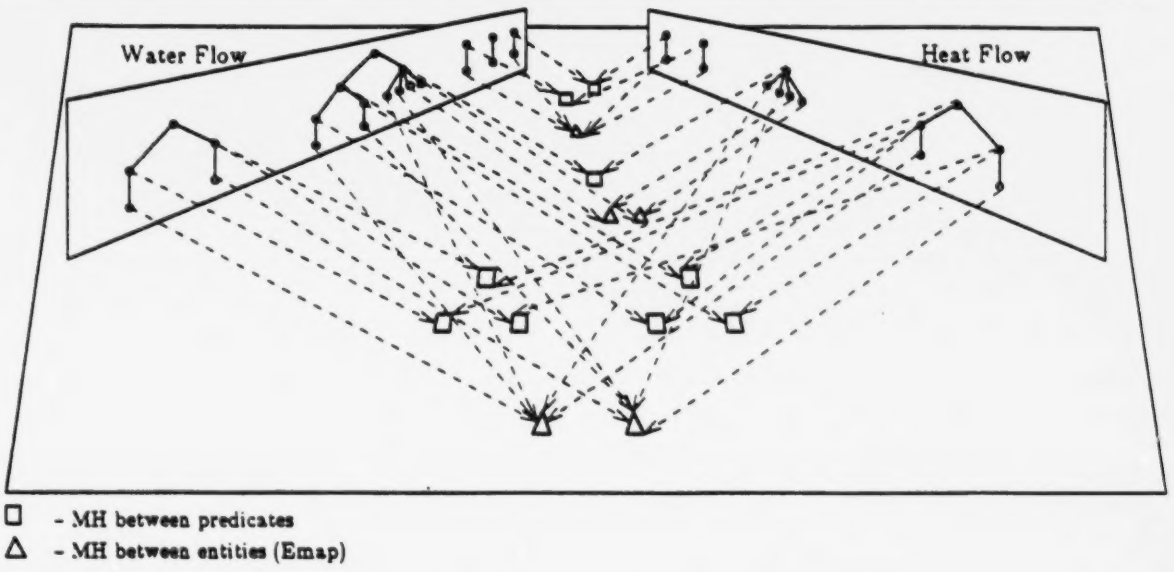



Figure 3: Local Match Construction. The water flow and heat flow descriptions of Figure 2 have been drawn in the abstract and placed to the left and right, respectively. The objects in the middle depict match hypotheses.

We will need some conventions for talking about match hypotheses. We denote the hypothesis that $b_i$ and $t_j$ match by $MH(b_i, t_j)$. When no ambiguity will result, we will simply say $MH$. The match hypotheses for a particular run of SME form a directed acyclic graph, with possibly many roots. We will use the same terminology to refer to the structural properties of graphs of match hypotheses (offspring, descendants, ancestors, root) as we use for describing items in Dgroups.

**Example: Simple analogy between heat and water** In this example we will use the literal similarity rule set, rather than true analogy, in order to better illustrate a variety of details of the algorithm. The result of running these rules on the water flow and heat flow dgroups of Figure 2 is shown in Figure 3 (see also Figure 4). Each match hypothesis locally pairs an item from the base dgroup with an item from the target dgroup.

There are several important things to notice in Figure 4. First, there can be more than one match hypothesis involving any particular base or target item. Here, TEMPERATURE can match with both DIAMETER and PRESSURE, since there are corresponding matches between the GREATER-THAN facts in both dgroups. Second, note that, with the exception of functions, predicates must match identically. Entities, on the other hand, are matched on the basis of their roles in the predicate structure. Thus while TEMPERATURE can match either PRESSURE or DIAMETER, IMPLIES cannot match anything but IMPLIES. This distinction reflects the fact that functions are often used to refer to objects and other types of entities, which are fair game for substitution under analogy. Third, not every possible correspondence is created. We do not, for example, attempt to match TEMPERATURE with water or heat with beaker. Local matches between entities are only created

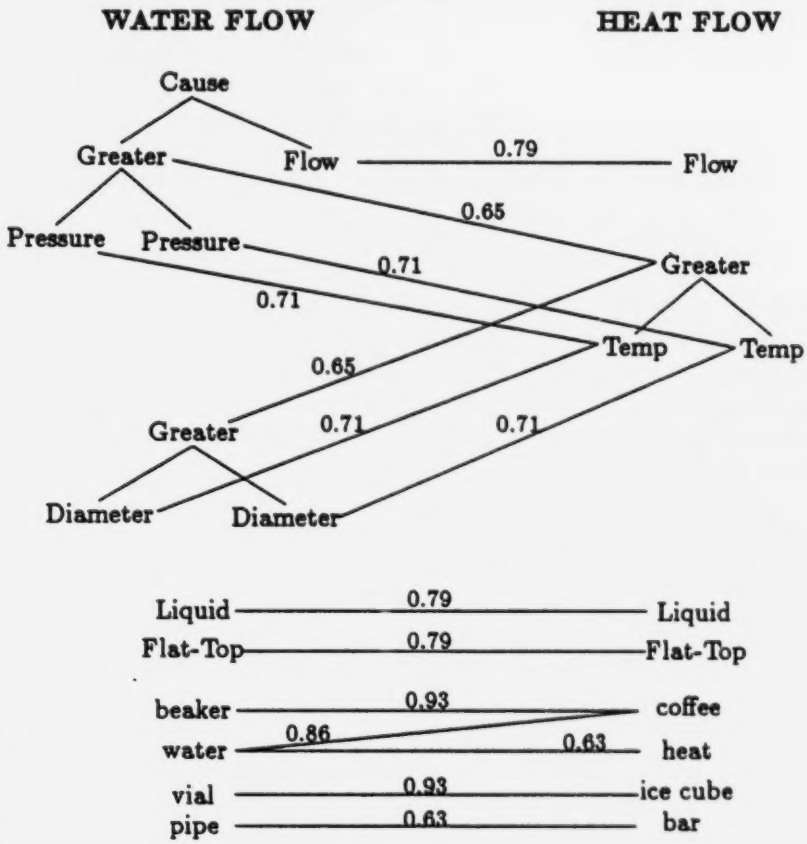



Figure 4: Water Flow / Heat Flow Analogy After Local Match Construction. The lines connecting water flow and heat flow items depict *match hypotheses*. Match hypotheses between entities are called *Emaps* (e.g., the link between beaker and coffee). Match hypotheses which are not descended from others are called *roots* (e.g., the links between the GREATER predicates and the link for the predicate FLOW). Each mapping receives a local evaluation score during subsequent processing, as described below

when justified by some other identity. This significantly constrains the number of possible matches in the typical case.

### 3.2.2 Step 2: Global Match Construction

The second step in the SME algorithm combines local match hypotheses into collections of global matches (Gmaps). Intuitively, each global match is the largest possible set of match hypotheses that depend on the same one to one object correspondences.

More formally, Gmaps consist of *maximal, structurally consistent* collections of match hypotheses. A collection of match hypotheses is structurally consistent if it satisfies two criteria:

1. *Preservation:* No two match hypotheses assign the same base item to multiple target items or any target item to multiple base items.

2. *Grounding:* If a match hypothesis MH is in the collection, then so are the match hypotheses which pair up all of the arguments of MH's base and target items.

The preservation criteria enforces strict one to one mappings. The grounding criteria preserves connected predicate structure. A collection is maximal if adding any additional match hypothesis would render the collection structurally inconsistent.

Requiring structural consistency both reduces the number of possible global collections and helps preserve the soundness and plausibility of the candidate inferences. Without it, every collection of local matches would need to be considered, wasting much effort on degenerate many-to-one mappings without any possible inferential value. The maximality condition also reduces the number of Gmaps, since otherwise every subset of a Gmap could itself be a Gmap.

The formation of global matches is composed of two primary stages:

1. *Compute consistency relationships:* Here we generate for each match hypothesis the sets of entity mappings it entails, what match hypotheses it locally conflicts with, and which match hypotheses it is structurally inconsistent with. This information simplifies the detection of contradictory sets of match hypotheses, a critical operation in the rest of the algorithm.

2. *Merge match hypotheses:* Compute Gmaps by successively combining match hypotheses as follows:

   (a) *Form initial combinations:* Combine interconnected and consistent match hypotheses into an initial set of Gmaps.

   (b) *Combine dependent Gmaps:* Since base and target dgroups are rarely isomorphic, some Gmaps in the initial set will overlap in ways that allow them to be merged. The advantage in merging them is that the new combination may provide structural support for candidate inferences.

   (c) *Combine independent collections:* The results of the previous step are next combined to form maximal consistent collections.

Importantly, the process of Gmap *construction* is completely independent of Gmap *evaluation*. Which Gmaps are constructed depends solely on structural consistency. Numerical evidence is used only as a source of information to compare the relative merits of Gmaps.

We now describe the algorithm in detail.

**Computing consistency relationships** Consistency checking is the crux of Gmap construction. Therefore we begin by defining several sets which represent, for each match hypothesis, which entity mappings it entails and which other hypotheses it is inconsistent with. Consider a particular match hypothesis $MH(b_i, t_j)$ involving base item $b_i$ and target item $t_j$. If $b_i$, $t_j$ are facts, then by the grounding criteria the match hypotheses linking their arguments must also be in any collection that $MH(b_i, t_j)$ is in. Applying this criteria recursively, we see that the match hypotheses indirectly mentioned by $MH(b_i, t_j)$ form a tree-like graph which "bottoms out" with match hypotheses involving primitive entities since they have no arguments (see Figure 5). This allows us to know what entity pairings a particular match hypothesis depends on:

**Definition 1.** An *Emap* is a match hypothesis between primitive entities. By recursive application of the grounding criteria, it can be seen that each match hypothesis implies a specific set of entity correspondences. Call the set of Emaps implied by a match hypothesis $MH(b_i, t_j)$ *Emaps*$(MH(b_i, t_j))$. *Emaps*$(MH(b_i, t_j))$ is simply the union of the Emaps supported by $MH(b_i, t_j)$'s descendants.

The preservation criteria enforces a strict one-to-one mapping. We must also associate with each $MH(b_i, t_j)$ the set of match hypotheses which conflict with it. The most obvious conflicts are those match hypotheses which provide alternate mappings for $b_i$ and $t_j$.

**Definition 2.** Given a match hypothesis $MH(b_i, t_j)$, the set *Conflicting*$(MH(b_i, t_j))$ consists of the set of match hypotheses which represent the alternate mappings for $b_i$ and $t_j$:

$$Conflicting(MH(b_i,\ t_j)) \equiv \bigcup_{b_k \in base} \{MH(b_k,\ t_j) \mid b_k \neq b_i\} \cup \bigcup_{t_k \in target} \{MH(b_i,\ t_k) \mid t_k \neq t_j\}$$

The set *Conflicting*$(MH(b_i, t_j))$ only notes local inconsistencies. However, we can use it and *Emaps*$(MH(b_i, t_j))$ to define a *NoGood* set which contains all match hypotheses that can never be in the same Gmap as $MH(b_i, t_j)$.

**Definition 3.** The *NoGood* set for a match hypothesis, $MH_i$, defines the set of match hypotheses which can never appear in the same Gmap as $MH_i$. This set is the union of $MH_i$'s *Conflicting* set with the *NoGood* sets for all of its descendents. If $MH_i$ is an Emap, then the *NoGood* set collapses into equalling its *Conflicting* set.

$$NoGood(MH_i) = Conflicting(MH_i) \cup \bigcup_{MH_j \in Args(MH_i)} NoGood(MH_j)$$

The algorithm computes *Conflicting*, *Emaps*, and *NoGood* sets as follows. First, *Conflicting* is computed for each match hypothesis, since it requires only local information. Second, *Emaps* and *NoGood* are computed for each Emap. Third, *Emaps* and *NoGood* sets are computed for all other match hypotheses by propagating the results from Emaps upwards to those match hypotheses which mention them.

Two observations should be made about this computation. First, these operations can be efficiently implemented via bit vectors. For example, SME assigns a unique bit position to each

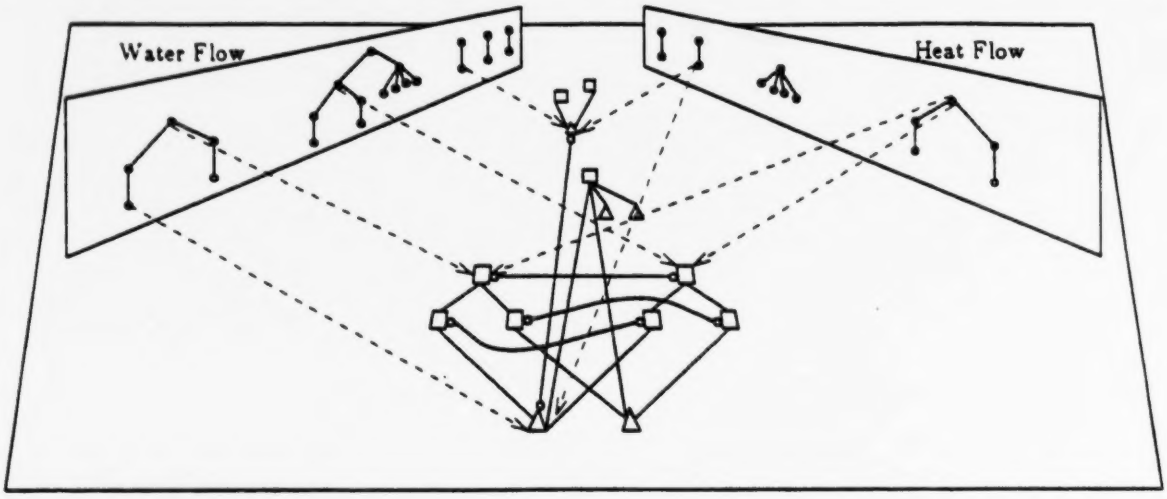



**Figure 5: Water Flow - Heat Flow analogy after computation of *Conflicting* relationships. Simple lines show the tree-like graph that the grounding criteria imposes upon match hypotheses. Lines with circular endpoints indicate the *Conflicting* relationships between matches. Some of the original lines from MH construction have been left in to show the source of a few *Conflicting* relations.**

match hypothesis, and carries out union and intersection operations by using OR and AND. Second, it is important to look for ***justification holes*** in the match hypothesis trees — match hypotheses whose arguments fail to match. Such match hypotheses will always fail the grounding criteria, and hence should be removed. For example, if one of the PRESSURE - TEMPERATURE match hypotheses had not been formed (see Figure 4), then the match between their governing GREATER predicates would be removed. Notice that removing justification holes eliminates many blatantly incorrect matches, such as trying to place an eighth-order IMPLIES in correspondence with a second-order IMPLIES.

The next step in Gmap construction is to identify those match hypotheses which are internally inconsistent, and thus cannot be part of any Gmap.

**Definition 4.** A match hypothesis is *Inconsistent* if the Emaps of one subtree of its descendants conflicts with the Emaps entailed by another subtree of its descendants:

$$Inconsistent(MH_i) \text{ iff } Emaps(MH_i) \cap NoGood(MH_i) \neq \emptyset$$

With the above information, SME can easily identify what combinations of match hypotheses are inconsistent. Furthermore, some match hypotheses can be identified as internally inconsistent, since it is possible for the descendants of a match hypothesis to have mutually incompatible bindings. Global match construction then proceeds by collecting sets of consistent match hypotheses. The maximality requirement on Gmaps suggests organizing the computation top-down, rather than

bottom-up. Call a match hypothesis which is not the descendant of any other match hypothesis a *root*. If a root is consistent, then the entire structure under it is consistent and may form an initial Gmap. However, base and target dgroups are rarely isomorphic, so several merge steps are also required. The next sections describe the procedure in detail.

**Merge Step 1: Form initial combinations.** The first step is to combine interconnected and consistent structures (Figure 6a). In the simplest case, the entire collection of descendants of a root may be collected together to form a globally consistent match. However, if the root is not consistent, then the same procedure is applied recursively to each descendant (i.e., each immediate descendant is now considered as a root). This effectively removes inconsistent match hypotheses from further consideration. The resulting set will be called $Gmaps_1$. The procedure is:

1. Let $Gmaps_1 = \emptyset$.
2. For every root $MH(b_i, t_j)$
   (a) If $\neg Inconsistent(MH(b_i, t_j))$, then create a Gmap $GM$ such that $Elements(GM) = Tree(MH(b_i, t_j))$.
   (b) If $Inconsistent(MH(b_i, t_j))$, then recurse on $Offspring(MH(b_i, t_j))$.
3. For every $GM \in Gmaps_1$,
   (a) $NoGood(GM) = \bigcup_{MH(b_i,t_j) \in Roots(GM)} NoGood(MH(b_i, t_j))$
   (b) $Emaps(GM) = \bigcup_{MH(b_i,t_j) \in Roots(GM)} Emaps(MH(b_i, t_j))$

At this stage inconsistent match hypotheses have been completely eliminated. If the base and target Dgroups had only a single root, the next two stages would have no effect. However, things are rarely that simple. Typically, elements of $Gmaps_1$ that are consistent with one another must be merged to provide maximality. Consistency between two GMaps is defined as:

$$Consistent(GMap_i \cup GMap_j) \text{ iff } \begin{array}{l} Elements(GMap_i) \cap NoGood(GMap_j) = \emptyset \\ \wedge \; NoGood(GMap_i) \cap Elements(GMap_j) = \emptyset \end{array}$$

**Merge Step 2: Combine dependent but unconnected Gmaps.** Because the target may lack some of the higher-order relations that exist in the base, two or more Gmaps from $Gmaps_1$ may be part of the same base structure. This step forms $Gmaps_2$ by merging all members of $Gmaps_1$ that share common base structure and that are consistent (Figure 6b). This puts all match hypotheses with shared base structure together. When candidate inferences are added to the Gmap, they will fill in the missing base structure.

**Merge Step 3: Combine independent collections.** Any two elements of $Gmaps_2$ which have overlapping base structure cannot consistently be merged, since if they could be then the previous step would have merged them. However, elements of $Gmaps_2$ which do *not* contain overlapping structure can be consistently merged, since they are independent from the perspective of structural consistency. This final step generates all consistent combinations of Gmaps from $Gmaps_2$ by successive unions, keeping only those combinations that are maximal (Figure 6c).

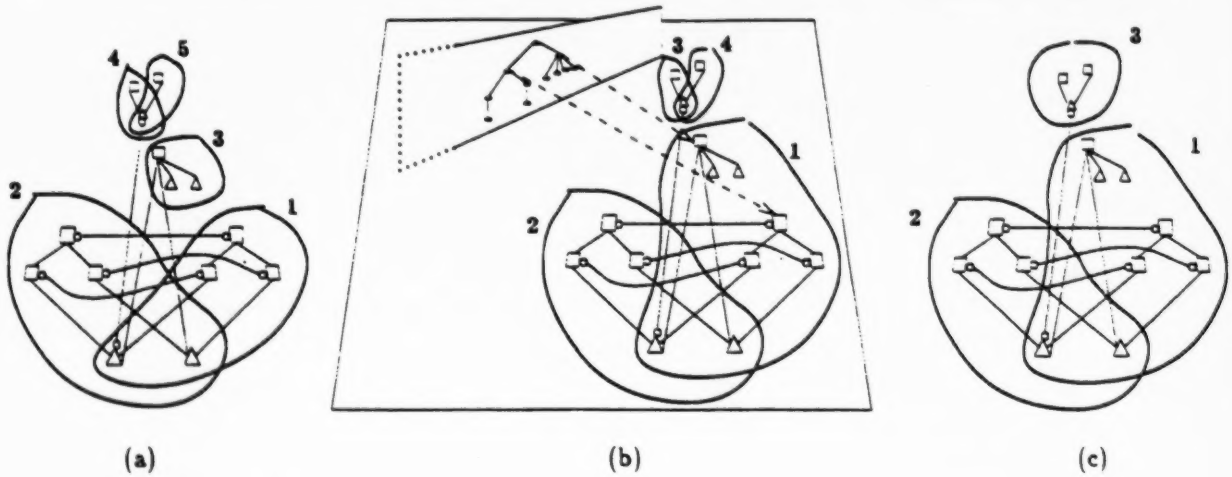



**Figure 6: GMap Construction. (a) Merge step 1: Interconnected and consistent. (b) Merge step 2: Consistent members of the same base structure. (c) Merge step 3: Any further consistent combinations.**

**Example: Simple analogy between heat and water** Figure 6 shows how the Gmaps are formed for the simple water flow / heat flow example. Recall that so far all we have is a large collection of match hypotheses, each representing a local pairing of an item from the base and an item from the target which could be part of a larger match. After merge step 1, only isolated collections exist. Merge step 2 combines the PRESSURE to TEMPERATURE mapping with the FLOW mapping. Finally, merge step 3 combines the isolated water and coffee attributes (see also Figure 7). Notice that the FLOW mapping is structurally consistent with the DIAMETER to TEMPERATURE mapping. However, because merge step 2 placed the FLOW mapping into the same Gmap as the PRESSURE to TEMPERATURE mapping, merge step 3 was unable to combine the FLOW mapping with the DIAMETER to TEMPERATURE Gmap.

### 3.2.3 Step 3: Compute Candidate Inferences

Each Gmap represents a set of correspondences that can serve as an interpretation of the match. For new knowledge to be generated about the target, there must be information from the base which can be "carried over" into the target. Not just any information can be carried over — it must be consistent with the substitutions imposed by the Gmap, and it must be *structurally grounded* in the Gmap. By structural grounding, we mean that its subexpressions must at some point intersect the base information belonging to the Gmap. Such structures form the *candidate inferences* of a Gmap.

Recall that Structure-Mapping does not guarantee that any candidate inference is valid. Each

candidate inference is only a surmise, which must be tested by other means. SME does guarantee that candidate inferences are structurally consistent and grounded. Aside from some simple structural consistency tests, mentioned below, SME uses no other rules of inference, or any other source of knowledge outside the language description and the base and target dgroups to evaluate a match. By theoretical assumption, such validity-checking is the province of other modules which use SME's output.[5]

To compute the candidate inferences for a Gmap $GM$, SME begins by examining each root $B_R$ in the base Dgroup to see if it is an ancestor of any base facts in the Gmap. If it is, then any elements in $Descendants(B_R)$ which are not in $BaseItems(GM)$ are included in the set of candidate inferences. A weak consistency check is performed by verifying that new, noncommutative facts are not commuted versions of existing facts. For example, if `(GREATER (MASS sun) (MASS planet))` existed in the target, `(GREATER (MASS planet) (MASS sun))` would be an inconsistent candidate inference.

The candidate inferences often include entities. Whenever possible, SME replaces all occurrences of base entities with their corresponding target entities. Sometimes, however, there will be base entities that have no corresponding target entity; i.e., the base entity is not part of any match hypothesis for that Gmap. What SME does depends on the type of entity. If the base entity is a constant, such as `zero`, it is brought directly into the target unchanged. Otherwise, SME introduces a new, hypothetical entity into the target which is represented as a skolem function of the original base entity. Such entities are represented as `(*skolem* base-entity)`.

**Example: Simple analogy between heat and water** We return now to our extended example. In Figure 7, Gmap #1 has the top level `CAUSE` predicate as its sole candidate inference. In other words, this Gmap suggests that the cause of the flow in the heat dgroup is the difference in temperatures.

Suppose the `FLOW` predicate was missing in the target Dgroup. Then the candidate inferences for a Gmap corresponding to the pressure inequality would be both `CAUSE` and `FLOW`, as well as conjectured target entities corresponding to water (heat) and pipe (bar). The two skolemized entities would be required because the match for the predicate `FLOW` is what provides a match from water and pipe to heat and bar. Note that `GREATER-THAN[DIAMETER(coffee), DIAMETER(ice cube)]` is not a valid candidate inference for the first Gmap because it does not intersect the existing Gmap structure.

### 3.2.4 Step 4: Compute Structural Evaluation Scores

Typically a particular pair of base and target will give rise to several Gmaps, each representing a different interpretation of the match. Often it is desired to select only a single Gmap, for example to represent the best interpretation of an analogy. Many of these evaluation criteria (including validity, usefulness, and so forth) lie outside the province of Structure-Mapping, and rely heavily on the domain and application. However, one important component of evaluation is *structural* — for example, one Gmap may be considered a better analogy than another if it embodies a more systematic match. SME provides a programmable mechanism for computing a *structural evaluation score* (SES) for each Gmap. This score can be used to rank-order the Gmaps in selecting the "best" analogy, or as a factor in a more complex (but external) evaluation procedure.

[5] One such module is described in [13,14].

```
Rule File: literal-similarity.rules     Number of Match Hypotheses: 14

Gmap #1: { (>PRESSURE  >TEMPERATURE)  (PRESSURE-BEAKER  TEMP-COFFEE)
           (PRESSURE-VIAL  TEMP-ICE-CUBE)  (WFLOW  HFLOW) }
    Emaps:  { (beaker coffee)  (vial ice-cube)  (water heat)  (pipe bar) }
    Weight:  5.99
    Candidate Inferences: (CAUSE >TEMPERATURE HFLOW)

Gmap #2: { (>DIAMETER  >TEMPERATURE)  (DIAMETER-1  TEMP-COFFEE)
           (DIAMETER-2  TEMP-ICE-CUBE) }
    Emaps:  { (beaker coffee)  (vial ice-cube) }
    Weight:  3.94
    Candidate Inferences:  { }

Gmap #3: { (LIQUID-3  LIQUID-5)  (FLAT-TOP-4  FLAT-TOP-6) }
    Emaps:  { (water coffee) }
    Weight:  2.44
    Candidate Inferences:  { }
```

Figure 7: Complete SME interpretation of Water Flow - Heat Flow Analogy.

The structural evaluation score is computed in two phases. First, each match hypothesis is assigned some local degree of evidence, independently of what Gmaps it belongs to. Second, the score for each Gmap is computed based on the evidence for its match hypotheses and the structural properties of the Gmap itself (such as the number and kind of candidate inferences). We will first introduce the general evidence processing architecture, followed by a description of how the structural evaluation scores are computed.

The scoring of matches is programmable: evidence for or against each aspect of the match is found by running *match evidence rules* and combining their results. These rules are similar syntactically to the match constructor rules and provide evidence scores for a match in the form of a probabilistic weight ranging between 0 and 1. Using rules to provide evidence greatly increases SME's programmability. Importantly, these evidence scores do *not* rely on any probabilistic or evidential information about the base or target per se. As we describe below, we currently are using the Dempster-Shafer formalism. However, our algorithms are independent of the details of Dempster-Shafer, and should work with any formalism for combining evidence.

The management of evidence rules is performed by a *Belief Maintenance System* (BMS) [12]. A BMS is a form of Truth-Maintenance system, extended to handle numerical weights for evidence and degree of belief. In SME, a BMS node is associated with every match hypothesis and Gmap. BMS justifications are Horn clauses, annotated with evidential weights. If the system knows that belief in $item_1$ is dependent in some way upon belief in $item_2$, then it automatically modifies belief in $item_1$ whenever new information causes a change in belief of $item_2$. In the limiting case of evidential weights of only 1 and 0 (i.e., true and false), the BMS behavior reduces to that of a standard justification-based TMS.

The current BMS implementation uses the Dempster-Shafer formalism for belief and combines

evidence with a simplified form of Dempster's rule of combination [54,48,33,12]. Using a modified form of Shafer's representation, we express the belief in some proposition by the pair (s(A), s(¬ A)), where s(A) represents the current amount of support for A and s(¬ A) is the current support against A. Dempster's rule provides a means for combining evidence based upon different sources of information. Thus, given that Belief(A)=(0.4, 0) and Belief(B)=(0.6, 0), together with (IMPLIES A C)$_{(0.8,0)}$ and (IMPLIES B C)$_{(1,0)}$, Dempster's rule provides a belief in C equal to (0.728, 0.0).

Pattern-directed rules are provided that trigger on certain events in the knowledge base (e.g., [45,6]). The rules are of the form:

(rule ((*NestedTriggers*)) (*Body*))

where a nested-trigger is of the form

((*Trigger*) (*Pattern*) [:test (*TestForm*)])

For example, the rule

```
(rule ((:intern (bird ?x)))
    (assert! (implies (bird ?x) (flies ?x) (0.90 . 0.05))))
```

causes the implication "if Fido is a bird, then there is a 90 to 95% probability that Fido can fly" to be asserted when (bird Fido) first appears in the knowledge base (whether it is believed or not). Notice that the rule system automatically converts the implies statement into a Horn clause; such translations will automatically be performed on all compositions of implies, and, or, and not.

We now describe the two phases for computing structural evaluation scores.

**Assigning local evidence** The local evidence for each match hypothesis is found by running the match evidence rules and combining their results. The rules trigger on various properties of a match hypothesis and provide evidence for or against it based upon those properties. In addition to providing direct evidence for a match hypothesis, the rules can also assert relationships between evidence for different hypotheses. For example, the systematicity preference for analogy is implemented by a rule that propagates belief in a match hypothesis to its offspring, thus increasing the evidence for Emaps that provide structural ground for a large systematic structure. All evidence is combined by asserting such relationships.

Let us consider some analogy evidence rules for concreteness. A match hypothesis involving two facts looks plausible if the predicates are the same, if the facts are of similar order in the base and target dgroups, and if their arguments can potentially match. A match hypothesis involving two facts appears implausible if the predicates are relations and their names are different or if the difference in their order is greater than one. For modularity, we separate each criteria into distinct rules. For example,

```
(assert! 'same-functor)
(rule ((:intern (MH ?b ?t) :test (and (fact? ?b) (fact? ?t)
                                      (eq (fact-functor ?b)
                                          (fact-functor ?t)))))
    (assert! (implies same-functor (MH ?b ?t) (0.5 . 0.0))))
```

states that "if the base item and target item are facts with the same functors, then supply 0.5 evidence in favor of the match hypothesis." A strong preference for systematicity is expressed by the local evidence rule:

```
(rule ((:intern (MH ?b1 ?t1))
       (:intern (MH ?b2 ?t2) :test (children-of? ?b2 ?t2 ?b1 ?t1)))
    (assert! (implies (MH ?b1 ?t1) (MH ?b2 ?t2) (0.8 . 0.0))))
```

which propagates 80% of a match hypothesis' belief to its offspring. As a result, the more matched structure that exists above a given match hypothesis, the more that hypothesis will be believed. Thus this "trickle down" effect provides a local encoding of the systematicity principle.

**Computing the global Structural Evaluation Score** Clearly an important factor in a Gmap's score is the evidence for the match hypotheses which participate in it. However, there are a number of other factors that are potentially relevant as well. Such factors include the number and size of connected components, the existence and structure of the candidate inferences, and other graph-theoretic properties of the Gmap.

We suspect that different factors will be relevant for different applications, and for modeling the different "analogical styles" of human subjects. Consequently, evidence rules are used to compute the evidence for Gmaps as well, to provide maximum flexibility. In this paper, we use only the following evidence rule for Gmaps:

```
(rule ((:intern (GMAP ?gmap)))
    (dolist (?mh (gmap-elements ?gmap))
        (assert! (implies ?mh (GMAP ?gmap)))))
```

which states that the belief of each match hypothesis is added to the belief of the Gmaps it is a member of. We have found this simple rule to be sufficient for all of the examples encountered so far, and so have not yet addressed the issue of evidence due to candidate inferences or graph theoretic structure.

Although the evidence for match hypotheses is constrained to be between 0 and 1, we do not normalize the evidence for Gmaps in the same way. Instead, we simply take as evidence for the Gmap the sum of the evidence for its match hypotheses[6]. Originally we combined evidence for Gmaps according to Dempster's rule, so that the sum of beliefs for all the Gmaps equal 1 [15]. We discovered two problems with this scheme. First, Dempster's rule is particularly susceptible to roundoff problems (i.e., unstable). Second, normalizing Gmap evidence prevents us from comparing matches using different base domains (as one would want to do for access experiments), since the score would be a function of the other Gmaps for a particular base and target pair. Under the current scheme, the evidence score can be used to compare matches of different base descriptions with the same target domain. However, it still cannot be used to compare two completely different analogies (i.e., different base, different target).[7]

[6]The BMS allows one to declare certain forms, such as (GMAP ?gmap), for special treatment. Thus, while the syntax of the Gmap evidence rule looks the same as the MH evidence rules (i.e., the use of INPLIES), evidence is combined by addition rather than Dempster's rule.

[7]One of our current research goals is the construction of a structural evaluator that would produce scores corresponding to a single, fixed scale. With this evaluator, SME would then be able to rate two completely different similarity matches as being equally good, regardless of how different their domain descriptions were in size.

The BMS justifications provide a valuable tool for exploring the consequences of evidence rules. By using interrogatives similar to those in truth-maintenance systems, one can get a clear picture of how the evidence rules are combining to produce the observed results. For example, the belief of (0.712, 0.0) in the match hypothesis between PRESSURE and TEMPERATURE (Figure 4) is explained by the BMS as follows:

```
(MH PRESSURE-BEAKER TEMP-COFFEE) has evidence (0.712, 0.0) due to
    IMPLICATION((MH >PRESSURE >TEMPERATURE))  (0.52, 0.0)
    IMPLICATION(CHILDREN-POTENTIAL)  (0.4, 0.0)
```

We suspect that the ability to "tune" the criteria for choosing a Gmap will be important for modeling individual differences in analogical style and a subject's domain knowledge. For example, a conservative strategy might favor taking Gmaps with some candidate inferences but not too many, in order to maximize the probability of being correct.

Although the match and evidence rules are programmable, it is important to note that all of the examples given in this paper use the same set of *analogy* rules, unless otherwise specified. In addition, the only other rules we have ever used are the *literal similarity* and *mere-appearance* rule sets. No "example-dependent" modifications have been performed to produce better results on any particular example.

**Example: Simple analogy between heat and water** Returning to Figure 7, note that the "strongest" interpretation (i.e., the one which has the highest structural evaluation score) is the one we would intuitively expect. In other words, beaker maps to coffee, vial maps to ice-cube, water maps to heat, pipe maps to bar, and PRESSURE maps to TEMPERATURE. Furthermore, we have the candidate inference that the temperature difference is what causes the flow.

### 3.3 Analysis

Here we review the SME algorithm and analyze its complexity. The algorithm is not straightforward, and depends critically on the particular data and match rules it is given. Consequently, we focus on identifying best-case and worst-case bounds as well as the factors which critically affect performance.

The algorithm is summarized in Figure 8. Referring back to our extended water flow / heat flow example, SME first hypothesized local matches between individual facts and entities. The resulting matches, shown in Figure 4, were then combined to form Gmaps. The three-step merging process, shown in Figure 6, produced the set of global mappings shown in Figure 7. The set of candidate inferences and a structural evaluation score was also created for each Gmap.

We use the following notation in the complexity analysis:

$\mathcal{E}_b \equiv$ Number of entities in the base dgroup.

$\mathcal{E}_t \equiv$ Number of entities in the target dgroup.

$\mathcal{F}_b \equiv$ Number of facts in the base dgroup.

$\mathcal{F}_t \equiv$ Number of facts in the target dgroup.

$\mathcal{M} \equiv$ Number of match hypotheses.

- **Run MHC rules to construct match hypotheses.**
- **Calculate the *Conflicting* set for each match hypothesis.**
- **Calculate the *EMaps* and *NoGood* sets for each match hypothesis by upward propagation from entity mappings.**
- **During the propagation, delete any match hypotheses that have justification holes.**
- **Merge match hypotheses into Gmaps.**
  1. **Interconnected and consistent.**
  2. **Consistent members of same base structure.**
  3. **Any further consistent combinations.**
- **Calculate the candidate inferences for each GMap.**
- **Score the matches**
  1. **Local match scores.**
  2. **Global structural evaluation scores.**

Figure 8: Summary of SME algorithm.

$\mathcal{G} \equiv$ Number of Gmaps

$N_b \equiv \mathcal{E}_b + \mathcal{F}_b$

$N_t \equiv \mathcal{E}_t + \mathcal{F}_t$

$N \equiv \frac{N_b + N_t}{2}$

### 3.3.1 Analysis of local match construction

The number of match rules is small, and hence largely irrelevant. The `:filter` rules are run for each pair of base and target predicates. Consequently, they will always require $O(N_b * N_t)$. The `:intern` rules are run once on each match hypothesis. In the worst case, $\mathcal{M} = N_b * N_t$, or roughly $N^2$. But in practice, the actual number of match hypotheses is substantially less, usually on the order of $cN$, where c is less than 5 and $N$ is the average of $N_b$ and $N_t$. Thus, in practice, `:intern` rules have a run time of approximately $O(N)$.

### 3.3.2 Analysis of *Conflicting* calculation

Recall that SME assigns a *Conflicting* set to each match hypothesis, $MH(b_i, t_j)$ which represents the alternate mappings for $b_i$ and $t_j$. The conflicting sets are calculated by examining the match hypotheses each base item appears in and the match hypotheses each target item appears in. Let $C$ be the average number of alternative matches each item in the base and target appears in. SME loops through the $C$ match hypotheses twice: once to form the bitwise union of these match hypotheses

and once to update each hypotheses' *Conflicting* set. Thus, the entire number of operations is

$$(\mathcal{F}_b * 2C) + (\mathcal{E}_b * 2C) + (\mathcal{F}_t * 2C) + (\mathcal{E}_t * 2C)$$

The worst case is when a match hypothesis is created between every base and target item. If we also assume $N_b = N_t$, then $C = N_t$ in that case. The number of operations becomes $4{N_t}^2$ or approximately $O(N^2)$. Conversely, the best case performance occurs when $C$ is 1, producing $O(max(N_b, N_t))$ operations. In our experiments so far, we find that $C$ is typically quite small, and so far has always been less than 10. Consequently, the typical performance lies between $O(N)$ and $O(N^2)$.

### 3.3.3 Analysis of *EMaps* and *NoGood* calculation

Recall that once the *Conflicting* sets are calculated, the *Emaps* and *NoGood* sets are propagated upwards from the entity mappings through the match hypotheses. By caching which $MH(b_i, t_j)$'s correspond to Emaps and using a queue, we only operate on each node once. Hence the worst and best case performance of this operation is $O(\mathcal{M})$.

### 3.3.4 Analysis of Gmap merge steps

Global matches are formed in a three step process. The first step is to collect all of the consistent connected components of match hypotheses. This requires starting down the match hypothesis roots, walking downwards to find consistent structures. Each tree walk takes $O(N_i)$, where $N_i$ is the number of nodes in the current tree. If there are $N_R$ roots, then the first merge step takes $O(N_R * N_i)$. Assuming that most of the match hypotheses will appear in only one or two trees (some roots may share substructure), we can approximate this by saying that the first merge step is $O(\mathcal{M})$. Call the number of partial Gmaps formed at this stage $\mathcal{G}_{P1}$.

The complexity of the previous steps has been, perhaps surprisingly, small. Matching computations usually have much worse performance, and we cannot completely escape this. In particular, a worst-case upper-bound for the second and third merge steps is $O(N!)$ (although worst-case for one implies best-case for the other).

The second merge step simply forms new Gmaps by combining those Gmaps from step one that intersect the same base structure. This is done by looping through each base description root, and checking which GMaps intersect the structure under that root. For each base description root, all possible consistent, maximal combinations of the Gmaps that intersect it are generated. In the worst case, every Gmap could intersect the same base structure. This would mean generating all possible consistent, maximal sets of Gmaps, which is the operation performed in step 3. We defer analysis of this case until then. In the other extreme, none of the Gmaps share a common base structure, and so step 2 requires $O({\mathcal{G}_{P1}}^2)$ operations, although this is not the best-case performance (see below). Typically, the second merge step is very quick and displays near best-case performance.

The final set of Gmaps is formed by generating all consistent combinations of Gmaps that exist after merge step 2. The complexity of this final merge step is directly related to the degree of "structuralness" of the base and target domains and how many different predicates are in use. Worst-case performance occurs when the description language is flat (i.e., no higher-order structure) and the same predicate occurs many times in both the base and the target. Consider a language with a single, unary predicate, and base and target dgroups each consisting of $N$distinct facts.

In this case every base fact can match with every target fact, and will suggest matching in turn the entities that serve as their arguments. This reduces to the problem of finding all isomorphic mappings between two equal size sets. There are $N!$ such mappings.

Now let us consider the best case. If the base and target dgroups give rise to a forest that has but one root, and that root is consistent, then there is only one Gmap! The second and third merge steps in this case are now independent of $N$, and are constant-time.

Of course, the typical case is somewhere between these two extremes. Typically the vocabulary of predicates is large, and the relationships between entities diverse. Structure provides a strong restriction on the number of possible interpretations for an analogy. By the time SME gets to this phase of the generation process, many of the match hypotheses have been filtered out as being structurally impossible. Others have been permanently bound to entire sets of match hypotheses due to merge steps 1 and 2. In addition, our match rules require that relational predicates only match other predicates with the same name. Thus, SME will perform badly on large dgroups with no structure. However, SME will perform extremely well on large dgroups with deep networks of diverse higher-order relationships. Semantically, the former case roughly corresponds to a jumble of unconnected facts, and the latter case to a complex argument or theory. The better organized and justified the knowledge, the better SME will perform.

While we reported potential $O(N!)$ performance for the second merge step, our experience shows that this is one of the fastest sections of the algorithm. We suspect that $O(N!)$ behavior for this step may only be possible in theory and would never show up in practice. This is because worst-case performance occurs when all members of $Gmaps_1$ intersect the same base-structure and so must be merged in all possible ways (as in step 3). However, Gmaps intersecting the same base structure are almost always consistent with one another, meaning that step 2 would usually merge $Gmaps_1$ into one Gmap in $O(\mathcal{G}_{P1})$ time!

### 3.3.5 Analysis of Finding Candidate Inferences

The candidate inferences are gathered by looping through the base description roots for each Gmap, collecting missing base facts whenever their structure intersects a match hypothesis in the Gmap. Each fact is tested to ensure that it is not already matched with part of the target description, or if it represents a contradiction of an existing target fact. The size of the typical candidate inference is inversely related to the number of roots: More roots implies less structure to infer, and vice versa. Thus in the worst case we have $O(\mathcal{G} * \mathcal{F}_b * \mathcal{F}_t)$, or roughly $O(N^4)$. However, this is an extreme worst-case. First, the $\mathcal{F}_t$ term implies that we check every target fact on each iteration. The algorithm actually only checks the pertinent target facts (i.e., those with the same functor), giving a more realistic worst-case of $O(N^3)$. In the best case, there will only be one Gmap and no candidate inferences, producing constant time behavior.

### 3.3.6 Analysis of Structural Evaluation Score computation

The complexity of the belief maintenance system is difficult to ascertain. However, it is more or less irrelevant, since it is possible to eliminate it if detailed justifications of evidential results are not needed. In the prototype version of SME [15], specialized evidence evaluation procedures were used which had most of the flexibility of the evidence rules, yet ran in $O(M)$ time.

While the flexibility the BMS provides is valuable, we note that in fact the majority of SME's processing time takes place within it – typically 70 to 80%. However, in terms of real-time this has

not yet been a serious limitation. On the examples in this paper (and most of the examples we have examined), SME runs in a matter of a few seconds on a Symbolics machine.

# 4 Examples

The Structure-Mapping Engine has been applied to over 40 analogies, drawn from a variety of domains and tasks. It is being used in psychological studies, comparing human responses with those of SME for both short stories and metaphors. It is also serving as a module in a machine learning program called PHINEAS, which uses analogy to discover and refine qualitative models of physical processes such as water flow and heat flow. Here we discuss a few examples to demonstrate SME's flexibility and generality.

## 4.1 Methodological constraints

Flexibility is a two-edged sword. The danger in using a program like SME is that one could imagine tailoring the match construction and evidence rules for each new example. Little would be learned by using the program in this way — we would have at best a series of "wind-up toys", a collection of ad-hoc programs which shed little theoretical light. Here we describe our techniques for avoiding tailorability.

First, all the cognitive simulation experiments were run with a fixed collection of rule sets (three sets to be exact). Each rule set represented a particular type of comparison sanctioned by the Structure-Mapping theory (*analogy*, *literal similarity*, and *mere appearance*). As described above, these rule sets are listed in Appendix A. The mere appearance rules (MA) care only about low-order matches: attributes and first-order relations. The analogy rules (TA) give strong preference to systems of relations and higher-order relations, while the literal similarity rules (LS) measure overall similarity. The first two examples in this section use the TA rules, while the last uses both TA and *MA* rules, as indicated.

While the choice of match construction rules is fixed by Structure-Mapping, the values of evidence weights are not. Although we have not performed a sensitivity analysis, in our preliminary explorations it appears that the Gmap rankings are not overly sensitive to the particular values of evidence weights. (Recall that which Gmaps are constructed is *independent* of the weights, and is determined only by the construction rules and structural consistency.)

Second, we have accumulated a standard description vocabulary which is used in all experiments. This is particularly important when encoding natural language stories, where the translation into a formal representation is underconstrained. By accumulating representation choices across stories, we attempt to free ourselves from "tailoring" particular examples.

Third, we have tested SME with descriptions generated automatically by other AI programs. A representation developed to perform useful inferences has fewer arbitrary choices than a representation developed specifically for learning programs. So far, we have used descriptions generated by two different qualitative simulation programs. The results are encouraging. For example, SME actually performs better on a water-flow / heat-flow comparison using more complex descriptions generated by GIZMO [19] than on many hand-generated descriptions.[8] We are working on other, similar systems, as described in Section 6.3.1.

[8] In fact, GIZMO stopped working before SME was built.

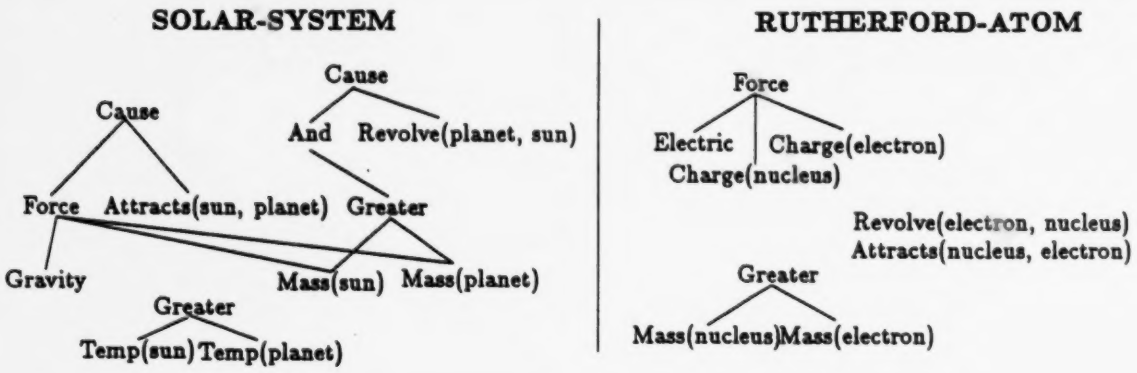



Figure 9: Solar System - Rutherford Atom Analogy.

## 4.2 Solar System - Rutherford Atom Analogy

The Rutherford model of the hydrogen atom was a classic use of analogy in science. A target system that was relatively unknown was explained in terms of the (relatively) well-understood behavior of the solar system. We illustrate SME's operation on this example with a simplified representation, shown in Figure 9.

The Structure-Mapping Engine constructed three possible interpretations. The highest-ranked mapping (SES = 6.03) pairs up the nucleus with the sun and the planet with the electron. This mapping is based on the mass inequality in the solar system playing the same role as the mass inequality in the atom. It sanctions the inference that the differences in masses, together with the mutual attraction of the nucleus and the electron, causes the electron to revolve around the nucleus. This is the standard interpretation of this analogy.

The other major Gmap (SES = 4.04) has the same entity correspondences, but maps the temperature difference between the sun and the planets onto the mass difference between the nucleus and the electron. The SES for this Gmap is low for two reasons. First, temperature and mass are different functions, and hence they receive less local evidence. The second reason is that, unlike the first Gmap, there is no systematic structure associated with temperature in the base dgroup, and thus other relations such as the match for ATTRACTS do not enter into this Gmap.

The third Gmap is a spurious collection of match hypotheses which imply that the mass of the sun should correspond to the mass of the electron, and the mass of the planet should correspond to the mass of the nucleus. There is even less higher-level structure to support this interpretation and so SES = 1.87.

This example demonstrates an important aspect of the Structure-Mapping account of analogy. Notice that the interpretation preferred on structural grounds is also the one with the most inferential import. This is not an accident; the systematicity principle captures the structural features of well-supported arguments. SME prefers interpretations based on a deep theory (i.e., a subset of a dgroup containing a system of higher-order relations) to those based on shallow associations (i.e., a subset of a dgroup with an assortment of miscellaneous facts).

**Water Flow History**

```
(Situation S0)
(Decreasing (Pressure (At beaker S0)))
(Increasing (Pressure (At vial S0)))
(Decreasing (Amount-of (At beaker S0)))
(Increasing (Amount-of (At vial S0)))
(Greater (Pressure (At beaker S0))
         (Pressure (At vial S0)))

(Situation S1)
(Meets S0 S1)
(Constant (Pressure (At beaker S1)))
(Constant (Pressure (At vial S1)))
(Constant (Amount-of (At beaker S1)))
(Constant (Amount-of (At vial S1)))
(Equal-To (Pressure (At beaker S1))
          (Pressure (At vial S1)))

(Function-Of (Pressure ?x)
             (Amount-of ?x))
```

**Heat Flow History**

```
(Situation S0)
(Decreasing (Temp (At horse-shoe S0)))
(Increasing (Temp (At water S0)))
(Greater (Temp (At horse-shoe S0))
         (Temp (At water S0)))

(Situation S1)
(Meets S0 S1)
(Constant (Temp (At horse-shoe S1)))
(Constant (Temp (At water S1)))
(Equal-To (Temp (At horse-shoe S1))
          (Temp (At water S1)))

(Function-Of (Temp ?x)
             (Heat ?x))
```

**Match**

| | | |
|---|---|---|
| Pressure | ↔ | Temperature |
| Amount-of | ↔ | Heat |
| S0 | ↔ | S0 |
| S1 | ↔ | S1 |
| beaker | ↔ | horse-shoe |
| vial | ↔ | water |

Figure 10: Analogical match between water flow history and heat flow history.

## 4.3 Discovering Heat Flow

The PHINEAS system is a program which learns by acting as a passive observer, relating observed physical phenomena to known theories of the world [13,14]. These theories are expressed as qualitative models of various physical processes, such as motion, boiling, and liquid flow, using Forbus' *Qualitative Process Theory* [17,18].

When PHINEAS is presented with a situation which the program's current models cannot explain, an analogical learning module is invoked which attempts to generate a new or revised model that can account for the new observation. This module uses SME in two ways. First, once an analogous experience has been accessed (i.e., one which exhibits similar behavior), SME is used to form a match between the changes observed in the prior experience and the changes taking place in the current situation. After these correspondences have been established, the domain models used to explain the previous experience are fetched and SME is used a second time to construct a new model for the

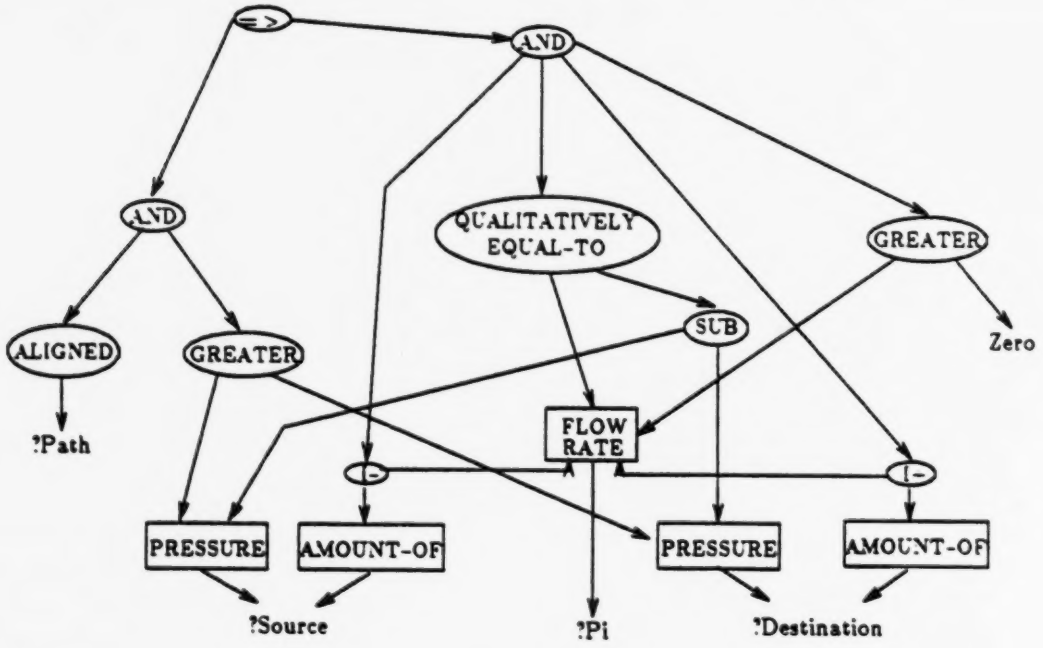



Figure 11: Qualitative Process Theory model of liquid flow.

current domain.

For example, suppose that the program was presented with measurements of the heat flow situation in Figure 1 and described in Figure 10. If the domain model does not include a theory of heat flow, PHINEAS will be unable to interpret the new observation.[9] Using SME, the program is able to establish an analogy with the previously encountered water flow experience shown in Figure 1 (see also Figure 10). This match serves to establish which properties from the two situations are behaving in the same way. As shown in Figure 10, the roles of the beaker and the vial in the water flow history are found to correspond to the roles of the horse shoe and water in the heat flow history, respectively. Those correspondences which provide a mapping between entities or between their quantities (e.g., Pressure and Temperature) are stored for later reference.

When it is satisfied that the chosen water flow history is sufficiently analogous to the current situation, PHINEAS fetches the relevant domain theory which led to its prior understanding of the base (water flow) experience. Figure 11 shows a domain description for water flow which has been used in Forbus' qualitative reasoning program [19,22]. This model (which is considerably more detailed than the liquid flow model shown previously) states that if we have an aligned fluid path between the beaker and the vial (i.e., the path either has no valves or if it does, they are all open), and the pressure in the beaker is greater than the pressure in the vial, then a fluid flow process

[9] PHINEAS uses the ATMI theory of measurement interpretation to explain observations. See [20] for details.

```
Gmap #1: { (AMOUNT-OF-35  HEAT-WATER)  (AMOUNT-OF-33  HEAT-HSHOE)
           (PRESSURE-BEAKER  TEMP-HSHOE) (PRESSURE-VIAL  TEMP-WATER) }
    Emaps:  { (beaker  horse-shoe)  (vial  water) }
    Weight:  2.675
    Candidate Inferences: (IMPLIES
                            (AND (ALIGNED (*skolem* pipe))
                                 (GREATER-THAN (A TEMP-HSHOE) (A TEMP-WATER)))
                            (AND (Q= (FLOW-RATE pi) (- TEMP-HSHOE TEMP-WATER))
                                 (GREATER-THAN (A (FLOW-RATE pi)) zero)
                                 (I+ HEAT-WATER (A (FLOW-RATE pi)))
                                 (I- HEAT-HSHOE (A (FLOW-RATE pi)))))
```

Figure 12: An Analogically Inferred Model of Heat Flow.

will be active. This process has a flow rate which is proportional to the difference between the two pressures. The flow rate has a positive influence on the amount of water in the vial and a negative influence on the amount of water in the beaker.

Using SME a second time, this theory is matched to the current heat flow situation, producing the output shown in Figure 12. The analogy at this stage is highly constrained, due to the set of entity and function correspondences established when the water flow and heat flow histories were matched. SME's rule-based architecture is critical to this operation: PHINEAS simply provides a new set of match constructor rules that only allow hypotheses consistent with the specific entity and function correspondences previously established. Entities and functions left without a match after the accessing stage are still left free to match other unmatched entities and functions. For example, the rule

```
(MHC-rule (:filter ?b ?t :test (sanctioned-pairing? (fact-functor ?b)
                                                    (fact-functor ?t)))
      (install-MH ?b ?t))
```

was used to force a match between those quantities which were found to be analogous during the behavioral similarity match (e.g., PRESSURE and TEMPERATURE) and prevent any alternate matches for these quantities (e.g., AMOUNT-OF and TEMPERATURE).

This example demonstrates several points. First, the "analogy" here is composed almost entirely of candidate inferences, since the system had no prior model of heat flow. Hence, the model was *constructed* by analogy rather than augmented by analogy. This shows the power of SME's candidate inference mechanism. Second, the example illustrates how SME's rule-based architecture supports situations in which the entity correspondences are given prior to the match, rather than derived as a result of the match. Finally, it shows the utility of introducing skolemized entities into the candidate inferences. The results produced by SME (Figure 12) contain the entity (*skolem* pipe). This indicates that, at the moment, the heat path is a conjectured entity. At this time, the system inspects its knowledge of paths to infer that immersion or physical contact is a likely heat path. However, we note that much knowledge gathering and refinement may still take place while leaving the heat path as a conjectured entity. An example of this strategy in the history of science is when *ether* was postulated to provide a medium for the flow of light waves.

**Base Story**

**Karla, an old Hawk, lived at the top of a tall oak tree. One afternoon, she saw a hunter on the ground with a bow and some crude arrows that had no feathers. The hunter took aim and shot at the hawk but missed. Karla knew that hunter wanted her feathers so she glided down to the hunter and offered to give him a few. The hunter was so grateful that he pledged never to shoot at a hawk again. He went off and shot deer instead.**

**Target Story - Analogy**

**Once there was a small country called Zerdia that learned to make the world's smartest computer.**

**One day Zerdia was attacked by its warlike neighbor, Gagrach. But the missiles were badly aimed and the attack failed. The Zerdian government realised that Gagrach wanted Zerdian computers so it offered to sell some of its computers to the country. The government of Gagrach was very pleased. It promised never to attack Zerdia again.**

**Target Story - Mere-Appearance**

**Once there was an eagle named Zerdia who donated a few of her tailfeathers to a sportsman so he would promise never to attack eagles.**

**One day Zerdia was nesting high on a rocky cliff when she saw the sportsman coming with a crossbow. Zerdia flew down to meet the man, but he attacked and felled her with a single bolt. As she fluttered to the ground Zerdia realised that the bolt had her own tailfeathers on it.**

Figure 13: Story Set Number 5.

## 4.4 Modelling Human Analogical Processing

SME is being used in several cognitive simulation studies. Our goal is to compare human responses with those of SME's for a variety of tasks and problems. For example, two psychological studies [29,50] have explored the variables that determine the *accessibility* of a similarity match and that determine the *inferential soundness* of a match. Structure-Mapping predicts that the degree of systematic relational overlap will determine soundness [25]. A further hypothesis suggested by Structure Mapping is that surface similarity will determine accessibility [29]. The psychological studies supported both hypotheses. In order to verify the computational assumptions we then ran SME on the same examples. Here we briefly summarize the methodology and the results — for details see [57].

The hypotheses were tested psychologically as follows. Pairs of short stories were constructed which were alike in one of the three ways: mere appearance, analogy, or literal similarity. Subjects first read a large set of stories. Then, in a second session, subjects saw similar stories and tried to retrieve the original stories (the access measure). After that, the subjects were then asked to judge the inferential soundness of each of the story pairs. For the cognitive simulation study, five groups of stories (15 in all) were encoded. Then pairs of stories were presented to SME, using different rule sets corresponding to analogy (the AN rules) and mere appearance (the MA rules). The results from the AN rules were used to estimate soundness, while the results from the MA rules were used to estimate accessibility. One of these story groups will be discussed in detail, showing how SME was used to simulate a test subject.

In the story set shown in Figure 13, the original story concerned a hawk named Karla who withstands an attack by a hunter. Two similar, target stories were used as potential analogies for the Karla narration. One was designed to be truly analogous (TA5) and describes a small country named Zerdia that withstands an attack by another country. The other story (MA5) was designed

**Analogical Match from *Karla* to *Zerdia the country (TA5).***

```
Rule File: appearance-match.rules   Number of Match Hypotheses: 12   Number of GMaps: 1

Gmap #1:
   (HAPPINESS-HUNTER HAPPINESS-GAGRACH) (ATTACK-HUNTER ATTACK-GAGRACH) (TAKE-FEATHERS BUY-SUPERCOMPUTER)
   (WARLIKE-HUNTER WARLIKE-GAGRACH) (DESIRE-FEATHERS DESIRE-SUPERCOMPUTER)
   (HAS-FEATHERS USE-SUPERCOMPUTER) (OFFER-FEATHERS OFFER-SUPERCOMPUTER) (WEAPON-BOW WEAPON-BOW)
 Emaps:  (KARLA1 ZERDIA12) (FEATHERS3 SUPERCOMPUTER14) (CROSS-BOW4 MISSILES15) (HUNTER2 GAGRACH13)
 Weight: 6.411572
```

**Analogical Match from *Karla* to *Zerdia the eagle (MA5).***

```
Rule File: appearance-match.rules   Number of Match Hypotheses: 14   Number of GMaps: 1

Gmap #1:
   (OFFER-FEATHERS OFFER-FEATHERS) (TAKE-FEATHERS TAKE-FEATHERS) (ATTACK-HUNTER ATTACK-SPORTSMAN)
   (SEE-KARLA SEE-ZERDIA) (HAS-FEATHERS HAS-FEATHERS) (BIRD-KARLA BIRD-ZERDIA) (WEAPON-BOW WEAPON-BOW)
   (DESIRE-FEATHERS DESIRE-FEATHERS) (WARLIKE-HUNTER WARLIKE-SPORTSMAN) (PERSON-HUNTER PERSON-SPORTSMAN)
 Emaps:  (FEATHERS3 FEATHERS9) (CROSS-BOW4 CROSS-BOW10) (HUNTER2 SPORTSMAN8) (KARLA1 ZERDIA7)
 Weight: 7.703568
```

Figure 14: SME's Analysis of Story Set 5, Using the MA Rules.

to be only superficially similar and describes an eagle named Zerdia who is killed by a sportsman. The representation of the Karla story given to SME was:

```
(CAUSE (EQUALS (HAPPINESS HUNTER) HIGH)
       (PROMISE HUNTER KARLA (NOT (ATTACK HUNTER KARLA))))
(CAUSE (OBTAIN HUNTER FEATHERS) (EQUALS (HAPPINESS HUNTER) HIGH))
(CAUSE (OFFER KARLA FEATHERS HUNTER) (OBTAIN HUNTER FEATHERS))
(CAUSE (REALIZE KARLA (DESIRE HUNTER FEATHERS))  (OFFER KARLA FEATHERS HUNTER))
(FOLLOW (EQUALS (SUCCESS (ATTACK HUNTER KARLA)) FAILED)
        (REALIZE KARLA (DESIRE HUNTER FEATHERS)))
(CAUSE (NOT (USED-FOR FEATHERS CROSS-BOW))  (EQUALS (SUCCESS (ATTACK HUNTER KARLA)) FAILED))
(FOLLOW (SEE KARLA HUNTER) (ATTACK HUNTER KARLA))
(WEAPON CROSS-BOW4)
(KARLAS-ASSET FEATHERS3)
(WARLIKE HUNTER2)
(PERSON HUNTER2)
(BIRD KARLA1)
```

To test the hypothesis that accessibility depends on the degree of surface match, SME was run on both pairs using the mere-appearance match rules. This measured their degree of superficial overlap and thus, according to our prediction, the relative likelihood of their accessing the original story. The output of SME for the MA task is given in Figure 14, which shows that the eagle story (SES = 7.7) has a higher mere-appearance match rating than the country story (SES = 6.4). Thus, if the surface-accessibility hypothesis is correct, the MA story "Zerdia the eagle" should have led

**Analogical Match from *Karla* to *Zerdia the country (TA5).***

```
Rule File: analogy.rules      Number of Match Hypotheses: 54      Number of GMaps: 1

Gmap #1:
   (CAUSE-PROMISE CAUSE-PROMISE) (SUCCESS-ATTACK SUCCESS-ATTACK) (HAPPY-HUNTER HAPPY-GAGRACH)
   (HAPPINESS-HUNTER HAPPINESS-GAGRACH) (REALIZE-DESIRE REALIZE-DESIRE) (CAUSE-TAKE CAUSE-BUY)
   (ATTACK-HUNTER ATTACK-GAGRACH) (DESIRE-FEATHERS DESIRE-SUPERCOMPUTER) (FAILED-ATTACK FAILED-ATTACK)
   (TAKE-FEATHERS BUY-SUPERCOMPUTER) (CAUSE-FAILED-ATTACK CAUSE-FAILED-ATTACK)
   (CAUSE-OFFER CAUSE-OFFER) (FOLLOW-REALIZE FOLLOW-REALIZE) (HAS-FEATHERS USE-SUPERCOMPUTER)
   (CAUSE-HAPPY CAUSE-HAPPY) (NOT-ATTACK NOT-ATTACK) (PROMISE-HUNTER PROMISE)
   (NOT-HAS-FEATHERS NOT-USE-SUPERCOMPUTER) (OFFER-FEATHERS OFFER-SUPERCOMPUTER)
 Emaps:  (HIGH23 HIGH17) (FEATHERS20 SUPERCOMPUTER14) (CROSS-BOW21 MISSILES15)
         (HUNTER19 GAGRACH13) (KARLA18 ZERDIA12) (FAILED22 FAILED16)
 Weight: 22.362718
```

**Analogical Match from *Karla* to *Zerdia the eagle (MA5).***

```
Rule File: analogy.rules      Number of Match Hypotheses: 47      Number of GMaps: 1

Gmap #1:
   (PROMISE-HUNTER PROMISE) (DESIRE-FEATHERS DESIRE-FEATHERS) (TAKE-FEATHERS TAKE-FEATHERS)
   (CAUSE-OFFER CAUSE-OFFER) (OFFER-FEATHERS OFFER-FEATHERS) (HAS-FEATHERS HAS-FEATHERS)
   (REALIZE-DESIRE REALIZE-DESIRE) (ATTACK-HUNTER ATTACK-SPORTSMAN) (NOT-ATTACK NOT-ATTACK)
   (SUCCESS-ATTACK SUCCESS-ATTACK) (FOLLOW-SEE-ATTACK FOLLOW-SEE) (SEE-KARLA SEE-ZERDIA)
   (FAILED-ATTACK SUCCESSFUL-ATTACK) (CAUSE-TAKE CAUSE-TAKE)
 Emaps:  (FAILED22 TRUE11) (KARLA18 ZERDIA7) (HUNTER19 SPORTSMAN8)
         (FEATHERS20 FEATHERS9) (CROSS-BOW21 CROSS-BOW10)
 Weight: 16.816530
```

Figure 15: SME's Analysis of Story Set 5, Using the TA Rules.

to greater accessibility to the original story for the human subjects than the TA story "Zerdia the country".

To test the determinants of soundness, SME was again run on the above pairs of stories, this time using the analogy (AN) match rules. The output of SME for the TA task is given in Figure 15. Notice that "Zerdia the country" (SES = 22.4) was found to be a better analogical match to the original Karla story than "Zerdia the eagle" (SES = 16.8). Thus, according to Gentner's systematicity principle, it should be judged more sound by human subjects.

The comparison of SME with human performance have been promising. For each of the five story sets examined, the preferences of SME qualitatively agreed with those of the human subjects. First, SME replicates human access patterns when in mere-appearance mode (MA rules). SME's rankings of surface similarity between the base and target are strongly correlated with accessibility of comparisons in people. Second, psychological evidence indicates that people do indeed use systematicity and consistency to rate the soundness of a match. As predicted, SME replicates subject's soundness rankings when in analogy mode (AN rules). These results help confirm that the similarity matches programmed into SME's AN and MA rules do a reasonable job of capturing

Table 1: SME performance on described examples.

| Example | Number base facts/entities | Number target facts/entities | # MH's | # Gmaps | Total BMS run time | Total match run time |
|---|---|---|---|---|---|---|
| Simple Water-Heat | 11/4 | 6/4 | 14 | 3 | 0.70 | 0.23 |
| Solar System-Atom | 12/2 | 9/2 | 16 | 3 | 0.91 | 0.28 |
| PHINEAS behavioral | 40/8 | 27/6 | 69 | 6 | 9.68 | 1.92 |
| PHINEAS theory | 19/11 | 13/6 | 10 | 1 | 0.17 | 0.66 |
| Base6-TA5 (AN) | 26/6 | 24/6 | 54 | 1 | 5.34 | 0.87 |
| Base6-MA5 (AN) | 26/6 | 24/5 | 47 | 1 | 4.55 | 0.98 |
| Base6-TA5 (MA) | 26/6 | 24/6 | 12 | 1 | 0.38 | 0.36 |
| Base6-MA5 (MA) | 26/6 | 24/5 | 14 | 1 | 0.73 | 0.46 |

NOTE: All times are given in seconds. Total match time is total SME run time minus BMS run time.

two different kinds of similarity matches that people implicitly use.

We make two additional observations. First, the results demonstrate the considerable leverage for cognitive modeling that SME's architecture provides. We know of no other general-purpose matcher which successfully simulates two *distinct* kinds of human similarity comparisons. Second, the short story analogies show that SME is capable of matching large structures as well as the smaller, simpler structures shown previously.

### 4.5 Review of Performance

SME is written in Common Lisp and the examples in this paper were run on a Symbolics 3640 with 8 megabytes resident memory. A variety of examples have been presented, representing a wide range of complexity. Table 1 shows a comparison of how well SME performed for each example. To give an accurate account of how fast SME is, we have separated the BMS run time from the total SME run time, since the computational cost of the BMS can be removed if additional inspectability is not needed. The run times are given in seconds. From these times it can be seen that SME is extremely fast at producing unevaluated Gmaps. In fact, it would seem to be close to linear in the number of match hypotheses and in the number of base and target facts. The majority of the run time is spent within the BMS, producing structural evaluation scores. However, the total run times are sufficiently short that we have opted to continue using the BMS for the time being, since it has proven to be a valuable analysis tool.

The longest runtime occurred for the behavioral match between the water flow and heat flow observations (PHINEAS behavioral). While the descriptions for this example were the largest, the primary source of slowdown was the flat representations used to describe the situations.

## 5 Comparison With Other Work

The Structure-Mapping theory has received a great deal of convergent theoretical support in artificial intelligence and psychology. Although there are differences in emphasis, there is now widespread agreement on the basic elements of one-to-one mappings of objects with carryover of predicates ([3,4,34,36,41,42,51,53,66]). Moreover, several of these researchers have adopted special cases of the systematicity principle. For example, Carbonell focuses on plans and goals as the high-order

relations that give constraint to a system, while Winston [65] focuses on causality. Structure-Mapping theory subsumes these treatments in three ways. First, it defines mapping rules which are independent of particular domains or primitives. Second, the Structure-Mapping characterization applies across a range of applications of analogy, including problem solving, understanding explanations, etc. Third, the Structure-Mapping account treats analogy as one of a family of similarity comparisons, each with particular psychological privileges, and thus explains more phenomena.

Some models have combined an explicit Structure-Mapping component to generate potential interpretations of a given analogy with a pragmatic component to select the relevant interpretation (e.g., [3,42]). Given our experience with PHINEAS, we believe SME will prove to be a useful tool for such systems.

SME computes a structural match first, and then uses this structural match to derive candidate inferences. The implementations of Winston [65,66] and Burstein [3] are similar to SME in this respect. An alternate strategy is used by Kedar-Cabelli [42] and Carbonell [4]. These programs do not perform a match *per se*, but instead attempt to carry over "relevant" structure first and modify it until it applies to the target domain. The match arises as an implicit result of the structure modification. We know of no complexity results available for this technique, but we suspect it is much worse than SME. It appears that there is great potential for extensive search in the modification method. Furthermore, the modification method effectively requires that the access mechanism is able to provide only salient structures (e.g., *purpose-directed* [42]), since the focusing mechanism of a partial match is not present. This means these systems are unlikely to ever derive a surprising result from an analogy.

A very different approach is taken by Holyoak [39]. In this account, there is no separate stage of structural matching. Instead, analogy is completely driven by the goals of the current problem-solving context. Retrieval of the base domain is driven by an abstract scheme of current problem-solving goals. Creating the mapping is interleaved with other problem-solving activities. This "pragmatic" account, while appealing in some ways, has several crucial limitations. First, the pragmatic model has no account of soundness. Without structural consistency, the search space for matching explodes (see below). Second, analogy is used for other purposes than problem solving. These purposes include contexts where relevance does not apply. Thus an analogy interpretation algorithm that requires relevance cannot be a general solution [26,27]. Analogy can be used to explain a new concept and to focus attention on a particular aspect of a situation. Analogy can result in noticing commonalities and conclusions that are totally irrelevant to the purpose at hand. Third, psychological data indicates that access is driven by surface similarity, not relevance, as described previously.

We believe the modularity imposed by the Structure-Mapping account has several desirable features over the pragmatic account. In the Structure-Mapping account, the same match procedure is used for all applications of analogy. For example, in a problem-solving environment, current plans and goals influence what is accessed. Once base and target are both present, the analogy mapping is performed, independently of the particular context. Its results can then examined and tested as part of the problem-solving process (see [26,27]).

SME demonstrates that an independent, structural matcher can be built which is useful in several tasks and for a variety of examples (over 40 at this writing). By contrast, no clear algorithms have been presented based on the pragmatic account, and published accounts so far [38] describe only two running examples. Another problem concerns potential complexity. We have not yet developed a good "typical case" analysis of SME's complexity, but empirical evidence indicates that it is quite

reasonable. The reason is that Structure-Mapping focuses on *local* properties of the representation. On the other hand, the pragmatic account appears to involve arbitrary inference, and arbitrary amounts of knowledge, in the mapping process. Thus we would expect that the average-case computational complexity of a pragmatically oriented matcher will actually be much worse than SME.

## 5.1 Matching Algorithms

To our knowledge, SME is unique in that it generates all structurally consistent analogical mappings without search. Previous matchers search through the space of possible matches and often return a single, best match (e.g., [11,35,43,46,60,61,64,65,66]). Some research on analogy has concluded that generating all possible interpretations is computationally intractable [35,65,43]. Our analysis and empirical results indicate that this conclusion must be substantially modified. Only when structural consistency does not exist, or is ignored, does the computation become intractable. For instance, in [43] the knowledge base was uniform and had no higher-order structure. In such cases exponential explosions are unavoidable.

Winston's matcher [65,66] heuristically searches for a single best match. It begins by enumerating all entity pairings and works upward to match relations, thus generating all $N_{Eb}!/(N_{Eb}\text{-}N_{Et})!$ possible entity pairings. Because SME only introduces entity pairings when suggested by potential shared relational structure, it typically generates many fewer entity pairings. Some limited amount of pruning due to domain-specific category information was also available on demand, such as requiring that males match with males. By contrast, SME ignores attributes when in analogy mode, unless they play a role in a larger systematic structure. Winston's scoring scheme would attribute one point for each shared relation (e.g., LOVE, CAUSE), property (e.g., STRONG, BEAUTIFUL), and class classification (e.g., A-KIND-OF(?x, woman)). Unlike SME's analogy rules, this scheme makes no distinction between a single, systematic relational chain and a large collection of independent facts.

Kline's RELAX system [43] focused on matching relations rather than entities. RELAX did not attempt to maintain structural consistency, allowing many-to-one mappings between entities or predicate instances. In conjunction with a semantic network, RELAX was able to match items having quite different syntax (e.g., (Segment A1 A2) matching (Angle A1 X A2)). However, there was no guarantee that the best match would be found due to local pruning during search.

Programs for forming inductive generalizations have also addressed the partial matching problem. These systems use a heuristically pruned search to build up sets of correspondences between terms which are then variablized to form generalized concept descriptions. Since these systems were not designed for analogy, they resemble the operation of SME programmed as a literal graph matcher (e.g., they could not match Pressure to Temperature). Hayes-Roth & McDermott's SPROUTER [35] and Dietrich & Michalski's INDUCE 1.2 [11] possess our restriction of one-to-one consistency in matching. Vere's THOTH system [60,61] uses less stringent match criteria. Once the initial sets of matched terms are built, previously unmatched terms may be added to the match if their constants are in related positions. In the process, THOTH may allow many–to–one mappings between terms.

The usefulness of many–to–one mappings in matches has been discussed in the literature [35,43]. Hayes-Roth & McDermott [35] advocate the need for many-to-one mappings among entities. Kline [43] calls for many-to-one mappings between propositions as well. For example, Kline points out that in trying to match a description of National League baseball to American

League baseball, the statement (`male NLpitcher`) should match both (`male ALpitcher`) and (`male ALdesignatedhitter`).

We disagree with this view, since we believe that structural consistency is central to analogy. Many–to–one mappings are permitted in artistic metaphor, but are not allowed in explanatory, predictive analogies [24,32]. However, we agree that multiple mappings are sometimes useful [8]. We propose to solve the problem by viewing many–to–one mappings as multiple analogies between the same base and target. Since SME produces all of the interpretations of an analogy, a postprocessor could keep more than one of them to achieve the beneficial effects of multiple interpretations, without sacrificing consistency and structural clarity. Thus, in the baseball example, SME would produce an *offense* interpretation and a *defense* interpretation. Postprocessing would combine these two Gmaps to form a single National League – American League analogy which mapped the National League pitcher into both the American League pitcher and American League designated hitter.

# 6 Discussion

We have described the Structure-Mapping Engine, a tool-kit for building matchers consistent with Gentner's Structure-Mapping theory of analogy and similarity. SME is both efficient and flexible. A particular matching algorithm is specified by a set of *constructor rules* and *evidence rules*. It produces all structurally consistent interpretations of a match, without backtracking. The interpretations include the candidate inferences suggested by the match and a structural evaluation score, giving a rough measure of quality. We believe we have described SME's algorithm in sufficient detail to allow successful replication by other interested researchers.

SME has been used both in cognitive simulation studies and a machine learning project. In the cognitive simulation studies, the results so far indicate that SME, when guided with analogy rules, replicates human performance. In the machine learning project (PHINEAS), SME's flexibility provides the means for constructing new qualitative theories to explain observations.

While our complexity analysis indicates that SME's worst-case performance is factorial, the empirical experience is that the typical behavior is much better than that. Importantly, the characteristic which determines efficiency is not size, but the *systematicity* of the knowledge being analyzed. Unlike many AI systems, the more systematic the knowledge, the better SME will perform.

In this section we discuss some broader implications of the project, and sketch some of our plans for future work.

## 6.1 Implications for representation

The SME algorithm is sensitive to the detailed form of the representation. It must be, since we are forbidding domain-specific inference in the matching process. Existing AI systems rarely have more than one or two distinct ways to describe any particular situation or theory. But as our programs grow more complex (or as we consider modeling the range and depth of human knowledge) the number of structurally distinct representations for the same situation is likely to increase. For example, a story might be represented at the highest level by a simple classification (i.e., `GREEK-TRAGEDY`), at an intermediate level by relationships involving the major characters (i.e., `(CAUSE (MELTING WAX) FALL23)`), and at the lowest level by something like conceptual dependencies. An engineer's knowledge of a calculator might include its functional description, the algorithms it uses, and the

axioms of arithmetic expressed in set theory. Unless there is some "window of overlap" between the levels of description for base and target, no analogy will be found. When our programs reach this complexity, how will SME cope with this situation?

There are several possible approaches to this problem. Consider the set of possible representations for a description. Assume these representations can be ordered (or at least partially ordered) in terms of degree of abstraction. If two descriptions are too abstract, there will be no predicate overlap (GREEK-TRAGEDY versus SHAKESPEARE-DRAMA). If two descriptions are greatly detailed, there will be too many spurious matches (e.g., describing the actions of characters every microsecond). The problem is to find levels of description which provide useful analogies. We believe one solution is to invoke SME repeatedly, using knowledge of the definitions of predicates to "slide" the base or target descriptions up or down in the space of possible representations appropriately.

An orthogonal consideration is the degree of systematicity. Worst-case behavior can occur frequently when representations are large and relatively flat. Changes in representation can make large differences. For example, a PHINEAS problem which took .. was reduced to .. by imposing more symbolic structure.

## 6.2 Addressing the Combinatorics

As we have shown, SME is $O(N^2)$ except for the last critical merge step, which has $O(N!)$ worst-case performance. Our experience has found that even moderately structural domain descriptions produce excellent performance. However, in practice it is not always convenient to avoid traditional, flat domain representations. For example, SME is unable to duplicate Kline's baseball analogy [43] within a reasonable amount of time (hours). This is due to his flat description of the domain (e.g., (MALE catcher), (BATS left-fielder), (BATS center-fielder), etc.). Thus for some cases, generating all possible interpretations of an analogy may be prohibitive. Previous work in this area has produced matching algorithms that are specifically designed around heuristic search mechanisms. SME offers a clean line between generating all possibilities and imposing heuristic limitations. If we stop after the first merge step, SME represents an $O(N^2)$ algorithm for generating the complete set of *minimal* Gmaps! The subsequent merge steps could then be heuristically driven through a limited search procedure (e.g., beam-search, best-first, etc.) to produce the best or N best maximal interpretations. Alternatively, we could retain the current SME design (recall that the second merge step is required to support candidate inference generation and is almost always $O(N^2)$ or better) and simply drop the troublesome third merge step. This is an (unused) option that the current implementation provides. We have not yet explored the ramifications of dropping merge step 3, although work with PHINEAS has shown the need for the maximality criterion in practice.

In the next sections, we discuss the potential for parallel versions of the SME algorithm. In particular, we argue that (1) there are many opportunities for parallel speedup, and (2) the expensive merge steps can be eliminated in principle.

### 6.2.1 Medium-grained Parallel Architectures

We begin by examining each stage of the algorithm to see how it might be decomposed into parallel operations, and what kinds of speedups might result.

**Constructing Match Hypotheses** Running the match constructor rules is a purely local, and can be broken down into one task for each pair of base item/target item which will either produce a hypothesis or not. With enough processors, this step becomes unit time.

**Computing *Conflicting*, *EMaps*, and *NoGood* sets** Computing the *Conflicting* set is a completely independent operation. It could either be organized around each base or target item, or around pairs of match hypotheses. Finding the *EMaps* and *NoGood* sets require propagation of results upwards, and hence must take time proportional to the maximum depth of the the match hypotheses.

**Merge Step 1: Form initial combinations** Recall that this step starts from the roots of the match hypothesis forest, adding the subtree to the list of Gmaps if the hypothesis is not inconsistent and recursing on its offspring otherwise. The results from each root are independent, and so may be done as separate tasks. If each recursive step spawns a new process to handle each offspring, then the minimum time is proportional again to the depth of the highest tree in the forest.

**Merge Step 2: Combine dependent but unconnected Gmaps** Recall that this step combines initial Gmaps which share common base structure and are not inconsistent when taken together. This procedure can be carried out bottom-up, merging pairs which are consistent and then iterating on the results. The computation time will be logarithmic in the number of Gmaps, assuming enough processors and ignoring allocation time.

**Merge Step 3: Combine independent collections** This can be performed like the previous step, but requiring that pairs of Gmaps not have common structure. Again, with enough processors the time is bounded by the log of the number of Gmaps. However, since the number of Gmaps is in the worst case factorial, the number of tasks required can become rather large.

The analysis above is cursory at best, and there are no doubt several problems lurking in creating a highly parallel version of the SME algorithm. However, we believe such algorithms could be very promising.

SME's simplicity also raises another interesting experimental possibility. Given that currently many medium-grain parallel computers are being built with reasonable amounts of RAM and a lisp environment on each machine, one can imagine simply loading a copy of SME into each processor. Access experiments, for example, would be greatly sped up by allowing a pool of SMEs to work over the knowledge base in a distributed fashion.

### 6.2.2 Connectionist Architectures

Another interesting approach would be to only generate a single, best Gmap while still maintaining SME's "no search" policy. The problem of choosing among all possible interpretations in analogy processing is very much like choosing among possible interpretations of the sentence "John shot two bucks" in natural language processing. A "no search" solution to this natural language problem was provided by the connectionist work of Waltz and Pollack [63]. In fact, it was their work in connectionist models of natural language processing that inspired certain aspects of the design of SME.

Connectionist architectures allow all possible answers to be implicitly represented at once [16,63]. Rather than explicitly constructing all possible sentence interpretations and then choosing the best one, Waltz and Pollack used their networks to implicitly represent all of the possible choices. Given a particular network, spreading activation and lateral inhibition were used to find the single best interpretation.

Consider the network produced by SME prior to the Gmap merge steps (shown in Figure 5). Some match hypotheses support each other (grounding criterion) while others inhibit each other (*Conflicting* relations). Viewing this as a spreading activation, lateral inhibition network, it appears that standard connectionist techniques could be used to produce the best interpretation without explicitly generating all Gmaps. Furthermore, it may be possible to generate the second-best, third-best, etc. interpretations on demand by inhibiting the nodes of the best interpretation, forcing the second best to rise. Thus SME would be able to establish a global interpretation simply as an indirect consequence of the establishment of local consistency. This would eliminate the single most expensive computation of the SME algorithm. By eliminating explicit generation of all Gmaps, the complexity of the algorithm could drop to the $O(N^2)$ required to generate the connectionist network.

## 6.3 Future Work

### 6.3.1 Cognitive Simulation

We have several cognitive simulation studies of analogical reasoning, memory, and learning involving SME in progress. We mention only one here. Psychological research shows a marked developmental shift in analogical processing. Young children rely on surface information in analogical mapping; at older ages, systematic mappings are preferred [30,31,40,62]. Further, there is some evidence that a similar shift from surface to systematic mappings occurs in the novice-expert transition in adults [7,44,51,52].

In both cases there are two very different interpretations for the analogical shift: (1) acquisition of knowledge; or (2) a change in the analogy algorithm. The knowledge-based interpretation is that children and novices lack the necessary higher-order relational structures to guide their analogizing. The second explanation is that the algorithm for analogical mapping changes, either due to maturation or learning. In human learning it is difficult to decide this issue, since exposure to domain knowledge and practice in analogy and reasoning tend to occur simultaneously. SME gives us a unique opportunity to vary independently the analogy algorithm and the amount and kind of domain knowledge. For example, we can compare identical evaluation algorithms operating on novice versus expert representations, or we can compare different analogy evaluation rules operating on the same representation. The performance of SME under these conditions can be compared with novice versus expert human performance.

We are also exploring ways to reduce the potential for tailorability in the process of translating descriptions provided as experimental stimuli for human subjects into formal representations for SME input. For example, Janice Skorstad is creating a graphical editor for producing graphical figures for experimental stimuli. One output of the editor is a printable picture, the other is a set of symbolic assertions with numerical parameters. The assertions are then passed into a simple inference engine (using a logic-based truth-maintenance system) which creates the relational structure SME requires by using rules to calculate interesting relationships, such as `INSIDE` or `LEFT-OF`.

Inspired by Winston's use of a pidgin-English parser for input [66], we are also seeking a

parser that, perhaps in conjunction with a simple inference engine, can produce useful descriptions of stories. Unfortunately, the parsers we have examined so far require input that is sufficiently restrictive that our subjects may find the stories unnatural.

### 6.3.2 Machine Learning Studies

While Falkenhainer's PHINEAS program is already capable of some learning behaviors, there is still more work to do. He is currently adding to its armatorium of knowledge-refinement techniques, and investigating access techniques appropriate for implementation on serial computers. He is also running PHINEAS on a number of additional examples to better test his theories.

PHINEAS is part of a larger machine learning project underway at University of Illinois, called the *Automated Physicist Project*. This project, spearheaded by Forbus and Gerald DeJong, also at Illinois, is building a collection of programs that use qualitative and quantitative techniques for reasoning and learning about the physical world. DeJong and his students have already built several interesting programs that use Explanation-Based Learning [9,10] to acquire knowledge of the physical world [55,49]. Forbus' group has already developed a number of useful qualitative reasoning programs [20,21,37] which can be used in learning projects (as PHINEAS demonstrates). By combining these results, we hope to build systems that can reason about a wide range of physical phenomena and learn both from observation and by being taught.

## 7 Acknowledgements

The authors wish to thank Steve Chien and Janice Skorstad for helpful comments on prior drafts of this paper. Janice Skorstad provided invaluable assistance in encoding domain models.

This research is supported by the Office of Naval Research, Contract No. N00014-85-K-0559. Additional support has been provided by IBM, both in the form of a Graduate Fellowship for Falkenhainer and a Faculty Development award for Forbus. The equipment used in this research was provided by an equipment grant from the Information Sciences Division of the Office of Naval Research, and from a gift from Texas Instruments.

## References

[1] Anderson, J., *The Architecture of Cognition*, Harvard University Press, Cambridge, Mass, 1983.

[2] Buckley, S., *Sun up to sun down*, McGraw-Hill Company, New York, 1979.

[3] Burstein, M., "Concept Formation by Incremental Analogical Reasoning and Debugging," *Proceedings of the Second International Workshop on Machine Learning*, University of Illinois, Monticello, Illinois, June, 1983. A revised version appears in *Machine Learning: An Artificial Intelligence Approach Vol. II*, R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (Eds.), Morgan Kaufman, 1986.

[4] Carbonell, J.G., "Learning by Analogy: Formulating and Generalizing Plans ¿From Past Experience," in *Machine Learning: An Artificial Intelligence Approach*, R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (Eds.), Morgan Kaufman, 1983.

[5] Carbonell, J.G., "Derivational Analogy in Problem Solving and Knowledge Acquisition," *Proceedings of the Second International Machine Learning Workshop*, University of Illinois, Monticello, Illinois, June, 1983. A revised version appears in *Machine Learning: An Artificial Approach Vol. II*, R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (Eds.), Morgan Kaufman, 1986.

[6] Charniak, E., C.K. Riesbeck, D.V. McDermott, *Artificial Intelligence Programming*, Lawrence Erlbaum and Associates, 1980.

[7] Chi, M.T.H., R. Glaser, E. Reese, Expertise in problem solving. In R. Sternberg (Ed.), *Advances in the psychology of human intelligence* (Vol. 1). Hillsdale, N.J., Erlbaum, 1982.

[8] Collins, A.M., & Gentner, D. How people construct mental models. In D. Holland and N. Quinn (Eds.) *Cultural models in language and thought*. Cambridge, England: Cambridge University, 1987.

[9] DeJong, G. Generalizations based on explanations. *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, August, 1981

[10] DeJong, G., and Mooney, R. Explanation-based Learning: An alternative view. *Machine Learning*, Volume 1, No. 2, 1986

[11] Diettrich, T., & Michalski, R.S., Inductive learning of structural descriptions: evaluation criteria and comparative review of selected methods, *Artificial Intelligence* **16**, 257-294, 1981.

[12] Falkenhainer, B., Towards a general-purpose belief maintenance system, *Proceedings of the Second Workshop on Uncertainty in AI*, Philadelphia, August, 1986a. Also Technical Report, UIUCDCS-R-87-1717, Department of Computer Science, University of Illinois.

[13] Falkenhainer, B., "An examination of the third stage in the analogy process: Verification-Based Analogical Learning," Technical Report UIUCDCS-R-86-1302, Department of Computer Science, University of Illinois, October, 1986b. A summary appears in *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, August, 1987.

[14] Falkenhainer, B., "Scientific theory formation through analogical inference", *Proceedings of the Fourth International Machine Learning Workshop*, Irvine, CA, June, 1987.

[15] Falkenhainer, B., K.D. Forbus, D. Gentner, The Structure-Mapping Engine, *Proceedings of the Fifth National Conference on Artificial Intelligence*, August, 1986.

[16] Feldman, J.A. & Ballard, D.H., Connectionist models and their properties, *Cognitive Science* **6**, 205-254, 1982.

[17] Forbus, K.D., "Qualitative Reasoning about Physical Processes", *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, August, 1981.

[18] Forbus, K.D., "Qualitative Process Theory," *Artificial Intelligence* **24**, 1984.

[19] Forbus, K.D., "Qualitative Process Theory," Technical Report No. 789, MIT Artificial Intelligence Laboratory, July, 1984.

[20] Forbus, K. "Interpreting measurements of physical systems", AAAI-86, August, 1986

[21] Forbus, K. "The Qualitative Process Engine" Technical Report No. UIUCDCS-R-86-1288, Department of Computer Science, University of Illinois, December, 1986

[22] Forbus, K.D. and D. Gentner, "Learning Physical Domains: Towards a Theoretical Framework," In *Proceedings of the Second International Machine Learning Workshop*, University of Illinois, Monticello, Illinois, June, 1983. A revised version appears in *Machine Learning: An Artificial Approach Vol. II*, R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (Eds.), Morgan Kaufmann, 1986.

[23] Gentner, D., *The Structure of Analogical Models in Science*, BBN Tech. Report No. 4451, Cambridge, MA., Bolt Beranek and Newman Inc., 1980.

[24] Gentner, D., Are Scientific Analogies Metaphors?, In Miall, D., *Metaphor: Problems and Perspectives*, Harvester Press, Ltd., Brighton, England, 1982.

[25] Gentner, D., Structure-Mapping: A Theoretical Framework for Analogy, *Cognitive Science* 7(2), 1983.

[26] Gentner, D., Mechanisms of analogy. To appear in S. Vosniadou and A. Ortony, (Eds.), *Similarity and analogical reasoning*. Presented in June, 1986.

[27] Gentner, D., Analogical Inference and Analogical Access, To appear in A. Preiditis (Ed.), *Analogica: Proceedings of the First Workshop on Analogical Reasoning*, London, Pitman Publishing Co. Presented in December, 1986.

[28] Gentner, D., & D.R. Gentner, Flowing Waters or Teeming Crowds: Mental Models of Electricity, In D. Gentner & A.L. Stevens, (Eds.), *Mental Models*, Erlbaum Associates, Hillsdale, N.J., 1983.

[29] Gentner, D., & R. Landers, Analogical Reminding: A Good Match is Hard to Find. In *Proceedings of the International Conference on Systems, Man and Cybernetics*. Tucson, Arizona, 1985.

[30] Gentner, D., & P. Stuart, Metaphor as Structure-Mapping: What Develops. Bolt Beranek and Newman Technical Report No. 5479, Cambridge, Massachusetts, 1983.

[31] Gentner, D., & C. Toupin, Systematicity and Surface Similarity in the Development of Analogy, *Cognitive Science*, 1986.

[32] Gentner, D., Falkenhainer, B., & Skorstad, J. Metaphor: The good, the bad and the ugly. *Proceedings of the Third Conference on Theoretical Issues in Natural Language Processing*, Las Cruces, New Mexico, January, 1987.

[33] Ginsberg, M.L., Non-Monotonic Reasoning Using Dempster's Rule, *Proceedings of the Fourth National Conference on Artificial Intelligence*, August, 1984.

[34] Greiner, S., Learning by understanding analogies, PhD Thesis (STAN-CS-1071), Department of Computer Science, Stanford University, September, 1985.

[35] Hayes-Roth, F., J. McDermott, An interference matching technique for inducing abstractions, *Communications of the ACM*, 21(5), May, 1978.

[36] Hofstadter, D.R., The Copycat project: An experiment in nondeterministic and creative analogies. M.I.T. A.I. Laboratory memo 755. Cambridge, Mass: M.I.T., 1984.

[37] Hogge, J. Compiling plan operators from domains expressed in qualitative process theory, *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, WA, July, 1987.

[38] Holland, J.H., Holyoak, K.J., Nisbett, R.E., & Thagard, P., *Induction: Processes of inference, learning, and discovery*, 1987.

[39] Holyoak, K.J. The pragmatics of analogical transfer. In G.H. Bower (Ed.), *The psychology of learning and motivation. Vol. I.* New York: Academic Press, 1984.

[40] Holyoak, K.J., E.N. Juin, D.O. Billman (in press). Development of analogical problem-solving skill. *Child Development.*

[41] Indurkhya, B., "Constrained Semantic Transference: A Formal Theory of Metaphors," Technical Report 85/008, Boston University, Department of Computer Science, October, 1985.

[42] Kedar-Cabelli, S., Purpose-Directed Analogy. *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, Irvine, CA, 1985.

[43] Kline, P.J., "Computing the similarity of structured objects by means of a heuristic search for correspondences", PhD Dissertation, Department of Psychology, University of Michigan, 1983.

[44] Larkin, J.H. Problem representations in physics. In D. Gentner & A.L. Stevens (Eds.) *Mental Models.* Hillsdale, N.J., Lawrence Erlbaum Associates, 1983.

[45] McAllester, D.A., "Reasoning Utility Package User's Manual," MIT AI Memo 667, April, 1980.

[46] Michalski, R.S., "Pattern recognition as rule-guided inductive inference" *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2(4), pp. 349-361, 1980.

[47] Novick, L.R. "Transfer and expertise in problem solving: A conceptual analysis." Stanford University: Unpublished manuscript, 1985.

[48] Prade, H., "A synthetic view of approximate reasoning techniques," *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, 1983.

[49] Rajamoney, S., DeJong, G., and Faltings, B. Towards a model of conceptual knowledge acquisition through directed experimentation. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, August, 1985.

[50] Rattermann, M.J., and Gentner, D. Analogy and Similarity: Determinants of accessibility and inferential soundness, *Proceedings of the Cognitive Science Society*, July, 1987.

[51] Reed, S.K., A structure-mapping model for word problems. Paper presented at the meeting of the Psychonomic Society, Boston, 1985.

[52] Ross, B.H., Remindings and their effects in learning a cognitive skill, *Cognitive Psychology*, **16**, 371-416, 1984.

[53] Rumelhart, D.E., & Norman, D.A., Analogical processes in learning. In J.R. Anderson (Ed.), *Cognitive skills and their acquisition*, Hillsdale, N.J., Erlbaum, 1981.

[54] Shafer, G., *A mathematical theory of evidence*, Princeton University Press, Princeton, New Jersey, 1976.

[55] Shavlik, J.W. Learning about momentum conservation. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, August, 1985

[56] Tversky, A. Representation of structure in similarity data: Problems and prospects. *Psychometrika* **39**, 373-421, 1974.

[57] Skorstad, J., Falkenhainer, B., Gentner, D., "Analogical Processing: A simulation and empirical corroboration", *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, WA, August, 1987.

[58] Van Lehn, K. & J.S. Brown, Planning nets: A representation for formalizing analogies and semantic models of procedural skills. In R.E. Snow, P.A. Federico & W.E. Montague (Eds.), *Aptitude, learning and instruction: Cognitive process analyses*. Hillsdale, N.J. Erlbaum, 1980.

[59] Van Lehn, K., "Felicity Conditions for Human Skill Acquisition: Validating an AI-based Theory," Xerox Palo Alto Research Center Technical Report CIS-21, 1983.

[60] Vere, S., "Induction of concepts in the predicate calculus", *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, 1975.

[61] Vere, S., "Inductive learning of relational productions", In Waterman & Hayes-Roth (Eds.), *Pattern-Directed Inference Systems*, 1978.

[62] Vosniadou, S., On the development of metaphoric competence. University of Illinois: Manuscript submitted for publication, 1985.

[63] Waltz, D.L. & Pollack, J.B., Massively Parallel Parsing: A strongly interactive model of natural language interpretation, *Cognitive Science* **9**, 51-74, 1985.

[64] Winston, P.H., Learning structural descriptions from examples, Ph.D. thesis, Report AI-TR-231, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, 1970.

[65] Winston, P.H., Learning and Reasoning by Analogy, *Communications of the ACM*, **23**(12), 1980.

[66] Winston, P.H., Learning new principles from precedents and exercises, *Artificial Intelligence*, **19**, 321-350, 1982.

# A SME Match Rules

The construction of a match is guided by a set of match rules that specify which facts and entities in the base and target might match and estimate the believability of each possible component of a match. In our experiments using SME, we currently use three types of rule sets, *literal similarity*, *analogy*, and *mere appearance*.

## A.1 Literal Similarity (LS) Rules

The *literal similarity* rules look at both relations and object descriptions.

```
;;;; Define MH constructor rules

;; If predicates are the same, match them

(MHC-rule (:filter ?b ?t :test (eq (fact-functor ?b) (fact-functor ?t)))
    (install-MH ?b ?t))

;; Intern rule for non-commutative predicates - corresponding arguments only.
;; Match compatible arguments of already matched items

(MHC-rule (:intern ?b ?t :test (and (fact? ?b) (fact? ?t)
                                    (not (commutative? (fact-functor ?b)))
                                    (not (commutative? (fact-functor ?t)))))
    (do ((bchildren (fact-arguments ?b) (cdr bchildren))
         (tchildren (fact-arguments ?t) (cdr tchildren)))
        ((or (null bchildren) (null tchildren)))
      (cond ((and (entity? (first bchildren)) (entity? (first tchildren)))
             (install-MH (first bchildren) (first tchildren)))
            ((and (function? (fact-functor (first bchildren)))
                  (function? (fact-functor (first tchildren))))
             (install-MH (first bchildren) (first tchildren))))))

;; Intern rule for commutative predicates - any "compatible" arguments, regardless of order.
;; Match compatible arguments of already matched items

(MHC-rule (:intern ?b ?t :test (and (fact? ?b) (fact? ?t)
                                    (commutative? (fact-functor ?b))
                                    (commutative? (fact-functor ?t))))
    (dolist (bchild (fact-arguments ?b))
      (dolist (tchild (fact-arguments ?t))
        (cond ((and (entity? bchild) (entity? tchild))
               (install-MH bchild tchild))
              ((and (function? (fact-functor bchild)) (function? (fact-functor tchild)))
               (install-MH bchild tchild))))))

;;;; Define MH evidence rules

;; having the same functor is a good sign
;;

(assert! same-functor)

(rule ((:intern (MH ?b ?t) :test (and (fact? ?b) (fact? ?t)
                                      (eq (fact-functor ?b) (fact-functor ?t)))))
   (if (function? (fact-functor ?b))
       (assert! (implies same-functor (MH ?b ?t) (0.2 . 0.0)))
       (assert! (implies same-functor (MH ?b ?t) (0.5 . 0.0)))))
```

```
;;check children (arguments) match potential
;;

(initial-assertion (assert! 'children-potential))

(rule ((:intern (MH ?b ?t) :test (and (fact? ?b) (fact? ?t))))
   (if (children-match-potential ?b ?t)
       (assert! (implies children-potential (MH ?b ?t) (0.4 . 0.0)))
       (assert! (implies children-potential (MH ?b ?t) (0.0 . 0.8)))))

;;if their order is similar, this is good. If the item is a function,
;; ignore since order comparisons give false support here.
;;

(initial-assertion (assert! 'order-similarity))

(rule ((:intern (MH ?b ?t) :test (and (fact? ?b) (fact? ?t)
                                      (not (function? (fact-functor ?b)))
                                      (not (function? (fact-functor ?t))))))
   (cond ((= (fact-order ?b) (fact-order ?t))
          (assert! (implies order-similarity (MH ?b ?t) (0.3 . 0.0))))
         ((or (= (fact-order ?b) (1+ (fact-order ?t)))
              (= (fact-order ?b) (1- (fact-order ?t))))
          (assert! (implies order-similarity (MH ?b ?t) (0.2 . 0.05))))))

;;propagate evidence down - systematicity
;; support for the arg will be 0.8 of the current support for the parent
;;

(rule ((:intern (MH ?b1 ?t1) :test (and (fact? ?b1) (fact? ?t1)))
       (:intern (MH ?b2 ?t2) :test (corresponding-arguments? ?b2 ?t2 ?b1 ?t1)))
   (assert! (implies (MH ?b1 ?t1) (MH ?b2 ?t2) (0.8 . 0.0))))

;;;; Gmap rules

;; Support from its MH's. At this time we ignore other factors such as number
;; of candidate inferences, etc.

(rule ((:intern (GMAP ?eg)))
   (dolist (mh (gm-elements ?eg))
      (assert! '(implies ,(mh-form mh) (GMAP ?eg)))))
```

## A.2 Analogy (AN) Rules

The *analogy* rules prefer systems of relations and discriminate against object descriptions. The analogy evidence rules are identical to the literal similarity evidence rules. The match constructor rules only differ in their check for attributes:

```
;;;; Define MH constructor rules

;; If predicates are the same, match them

(MHC-rule (:filter ?b ?t :test (and (eq (fact-functor ?b) (fact-functor ?t))
                                    (not (attribute? (fact-functor ?b)))))
    (install-MH ?b ?t))
```

```
;; Match compatible arguments of already matched items.
;; Notice attributes are allowed to match here, since they are part of some higher relation that matched.

;; Intern rule for non-commutative predicates - corresponding arguments only.

(MHC-rule (:intern ?b ?t :test (and (fact? ?b) (fact? ?t)
                                    (not (commutative? (fact-functor ?b)))
                                    (not (commutative? (fact-functor ?t)))))
    (do ((bchildren (fact-arguments ?b) (cdr bchildren))
         (tchildren (fact-arguments ?t) (cdr tchildren)))
        ((or (null bchildren) (null tchildren)))
      (cond ((and (entity? (first bchildren)) (entity? (first tchildren)))
             (install-MH (first bchildren) (first tchildren)))
            ((and (function? (fact-functor (first bchildren)))
                  (function? (fact-functor (first tchildren))))
             (install-MH (first bchildren) (first tchildren)))
            ((and (attribute? (fact-functor (first bchildren)))
                  (eq (fact-functor (first bchildren)) (fact-functor (first tchildren))))
             (install-MH (first bchildren) (first tchildren))))))

;; Intern rule for commutative predicates - any "compatible" arguments, not necessarily corresponding.

(MHC-rule (:intern ?b ?t :test (and (fact? ?b) (fact? ?t)
                                    (commutative? (fact-functor ?b))
                                    (commutative? (fact-functor ?t))))
    (dolist (bchild (fact-arguments ?b))
      (dolist (tchild (fact-arguments ?t))
        (cond ((and (entity? bchild) (entity? tchild))
               (install-MH bchild tchild))
              ((and (function? (fact-functor bchild))
                    (function? (fact-functor tchild)))
               (install-MH bchild tchild))
              ((and (attribute? (fact-functor bchild))
                    (eq (fact-functor bchild) (fact-functor tchild)))
               (install-MH bchild tchild))))))
```

## A.3 Mere Appearance (MA) Rules

The *mere appearance* rules focus on object descriptions and prevent matches between functions or relations. As a result, the number of evidence rules is greatly reduced.

```
;;;; Define MH constructor rules

(MHC-rule (:filter ?b ?t :test (and (eq (fact-functor ?b) (fact-functor ?t))
                                    (<= (fact-order ?b) 1)
                                    (<= (fact-order ?t) 1)))
    (install-MH ?b ?t))

(MHC-rule (:intern ?b ?t :test (and (fact? ?b) (fact? ?t)
                                    (not (commutative? (fact-functor ?b)))
                                    (not (commutative? (fact-functor ?t)))))
    (do ((bchildren (fact-arguments ?b) (cdr bchildren))
         (tchildren (fact-arguments ?t) (cdr tchildren)))
        ((or (null bchildren) (null tchildren)))
      (if (and (entity? (first bchildren)) (entity? (first tchildren)))
```

```
          (install-MH (first bchildren) (first tchildren)))))

(MHC-rule (:intern ?b ?t :test (and (fact? ?b) (fact? ?t)
                                    (commutative? (fact-functor ?b))
                                    (commutative? (fact-functor ?t))))
    (dolist (bchild (fact-arguments ?b))
      (dolist (tchild (fact-arguments ?t))
        (if (and (entity? bchild) (entity? tchild))
           (install-MH bchild tchild)))))

;;;; Define MH evidence rules

;; having the same functor is a good sign
;;

(initial-assertion (assert! 'same-functor))

(rule ((:intern (MH ?b ?t) :test (and (fact? ?b) (fact? ?t)
                                      (eq (fact-functor ?b) (fact-functor ?t)))))
       (cond ((attribute? (fact-functor ?b))
              (assert! (implies same-functor (MH ?b ?t) (0.5 . 0.0))))
             ((= 1 (max (fact-order ?b) (fact-order ?t)))
              (assert! (implies same-functor (MH ?b ?t) (0.4 . 0.0))))))

;;propagate evidence down - only for entity MH's caused by attribute pairings
;; support for the arg will be 0.9 of the current support for the parent
;;

(rule ((:intern (MH ?b1 ?t1) :test (and (fact? ?b1) (fact? ?t1)
                                        (<=  (max (fact-order ?b1)(fact-order ?t1))  1)))
       (:intern (MH ?b2 ?t2) :test (corresponding-arguments? ?b2 ?t2 ?b1 ?t1)))
    (assert! (implies (MH ?b1 ?t1) (MH ?b2 ?t2) (0.9 . 0.0))))

;;;; Gmap rules

;;; Support from its MH's. At this time we ignore other factors such as number of candidate inferences

(rule ((:intern (GMAP ?eg)))
   (dolist (mh (gm-elements ?eg))
      (assert! '(implies ,(mh-form mh) (GMAP ?eg)))))
```

# B Sample Domain Descriptions

In this section we show the domain descriptions given to SME for the described examples.

## B.1 Simple Water Flow - Heat Flow

### Water Flow

```
(defEntity water :type inanimate)
(defEntity beaker :type inanimate)
(defEntity vial :type inanimate)
(defEntity pipe :type inanimate)
```

```
(defDescription simple-water-flow
    entities (water beaker vial pipe)
    facts  (((flow beaker vial water pipe) :name wflow)
            ((pressure beaker) :name pressure-beaker)
            ((pressure vial)   :name pressure-vial)
            ((greater pressure-beaker pressure-vial) :name >pressure)
            ((greater (diameter beaker) (diameter vial)) :name >diameter)
            ((cause >pressure wflow) :name cause-flow)
            (flat-top water)
            (liquid water)))
```

### Heat Flow

```
(defEntity coffee :type inanimate)
(defEntity ice-cube :type inanimate)
(defEntity bar :type inanimate)
(defEntity heat :type inanimate)

(defDescription simple-heat-flow
    entities (coffee ice-cube bar heat)
    facts  (((flow coffee ice-cube heat bar) :name hflow)
            ((temperature coffee) :name temp-coffee)
            ((temperature ice-cube) :name temp-ice-cube)
            ((greater temp-coffee temp-ice-cube) :name >temperature)
            (flat-top coffee)
            (liquid coffee)))
```

## B.2 Solar-System - Rutherford Atom

### Solar System

```
(defEntity sun :type inanimate)
(defEntity planet :type inanimate)

(defDescription solar-system
    entities (sun planet)
    facts  (((mass sun) :name mass-sun)
            ((mass planet) :name mass-planet)
            ((greater mass-sun mass-planet) :name >mass)
            ((attracts sun planet) :name attracts)
            ((revolve-around planet sun) :name revolve)
            ((and >mass attracts) :name and1)
            ((cause and1 revolve) :name cause-revolve)
            ((temperature sun) :name temp-sun)
            ((temperature planet) :name temp-planet)
            ((greater temp-sun temp-planet) :name >temp)
            ((gravity mass-sun mass-planet) :name force-gravity)
            ((cause force-gravity attracts) :name why-attracts)))
```

### Rutherford Atom

```
(defEntity nucleus :type inanimate)
(defEntity electron :type inanimate)
```

```
(defDescription rutherford-atom
    entities (nucleus electron)
    facts  (((mass nucleus) :name mass-n)
            ((mass electron) :name mass-e)
            ((greater mass-n mass-e) :name >mass)
            ((attracts nucleus electron) :name attracts)
            ((revolve-around electron nucleus) :name revolve)
            ((charge electron) :name q-electron)
            ((charge nucleus) :name q-nucleus)
            ((opposite-sign q-nucleus q-electron) :name >charge)
            ((cause >charge attracts) :name why-attracts)))
```

## B.3 Karla Stories

### Zerdia the eagle - base story

```
(defEntity Karla)
(defEntity hunter)
(defEntity feathers)
(defEntity cross-bow)
(defEntity Failed)
(defEntity high)

(defDescription base-5
    entities  (Karla hunter feathers cross-bow Failed high)
    facts  (((bird Karla)     :name bird-Karla)
            ((person hunter) :name person-hunter)
            ((warlike hunter)         :name warlike-hunter)
            ((Karlas-asset feathers) :name feathers-asset)
            ((weapon cross-bow)       :name weapon-bow)
            ((used-for feathers cross-bow ) :name has-feathers)
            ((not has-feathers)       :name not-has-feathers)
            ((attack hunter Karla)    :name attack-hunter)
            ((not attack-hunter)      :name not-attack)
            ((see Karla hunter)       :name see-Karla)
            ((follow see-Karla attack-hunter) :name follow-see-attack)
            ((success attack-hunter) :name success-attack)
            ((equals success-attack Failed)        :name failed-attack)
            ((cause not-has-feathers failed-attack)  :name cause-failed-attack)
            ((desire hunter feathers)         :name desire-feathers)
            ((realize Karla desire-feathers) :name realize-desire)
            ((follow failed-attack realize-desire)     :name follow-realize)
            ((offer Karla feathers hunter)    :name offer-feathers)
            ((cause realize-desire offer-feathers)    :name cause-offer)
            ((obtain hunter feathers)  :name take-feathers)
            ((cause offer-feathers take-feathers)     :name cause-take)
            ((happiness hunter)       :name happiness-hunter)
            ((equals happiness-hunter high)  :name happy-hunter)
            ((cause take-feathers happy-hunter)       :name cause-happy)
            ((promise hunter Karla not-attack)        :name promise-hunter)
            ((cause happy-hunter promise-hunter)     :name cause-promise)))
```

### Zerdia the country - TA5

```
(defEntity Zerdia)
```

```
(defEntity Gagrach)
(defEntity supercomputer)
(defEntity missiles)
(defEntity failed)
(defEntity high)

(defDescription ta-5
    entities   (Zerdia Gagrach supercomputer missiles failed high)
    facts  (((country Zerdia)          :name country-Zerdia)
            ((country Gagrach)         :name country-Gagrach)
            ((warlike Gagrach)         :name warlike-Gagrach)
            ((Zerdias-asset supercomputer)   :name supercomputer-asset)
            ((weapon missiles)         :name weapon-bow)
            ((used-for supercomputer missiles ) :name use-supercomputer)
            ((not use-supercomputer) :name not-use-supercomputer)
            ((attack Gagrach Zerdia) :name attack-Gagrach)
            ((not attack-Gagrach)    :name not-attack)
            ((success attack-Gagrach)          :name success-attack)
            ((equals success-attack failed)        :name failed-attack)
            ((cause not-use-supercomputer failed-attack) :name cause-failed-attack)
            ((desire Gagrach supercomputer)  :name desire-supercomputer)
            ((realize Zerdia desire-supercomputer)   :name realize-desire)
            ((follow failed-attack realize-desire)    :name follow-realize)
            ((offer Zerdia supercomputer Gagrach)    :name offer-supercomputer)
            ((cause realize-desire offer-supercomputer) :name cause-offer)
            ((obtain Gagrach supercomputer)     :name buy-supercomputer)
            ((cause offer-supercomputer buy-supercomputer)  :name cause-buy)
            ((happiness Gagrach)     :name happiness-Gagrach)
            ((equals happiness-Gagrach high) :name happy-Gagrach)
            ((cause buy-supercomputer happy-Gagrach) :name cause-happy)
            ((promise Gagrach Zerdia not-attack)     :name promise)
            ((cause happy-Gagrach promise)   :name cause-promise)))
```

**Zerdia the hawk - MA5**

```
(defEntity Zerdia)
(defEntity sportsman)
(defEntity feathers)
(defEntity cross-bow)
(defEntity true)

(defDescription ma-5
    entities   (Zerdia sportsman feathers cross-bow true)
    facts  (((bird Zerdia)   :name bird-Zerdia)
            ((person sportsman)        :name person-sportsman)
            ((warlike sportsman)       :name warlike-sportsman)
            ((Zerdias-asset feathers) :name feathers-asset)
            ((weapon cross-bow)        :name weapon-bow)
            ((used-for feathers cross-bow )  :name has-feathers)
            ((desire sportsman feathers)     :name desire-feathers)
            ((realize Zerdia desire-feathers)       :name realize-desire)
            ((offer Zerdia feathers sportsman)      :name offer-feathers)
            ((cause realize-desire offer-feathers) :name cause-offer)
            ((obtain sportsman feathers)            :name take-feathers)
            ((cause offer-feathers take-feathers)  :name cause-take)
            ((attack sportsman Zerdia)      :name attack-sportsman)
```

```
((not attack-sportsman)          :name not-attack)
((promise sportsman Zerdia not-attack)   :name promise)
((cause take-feathers promise) :name cause-promise)
((see Zerdia sportsman)          :name see-Zerdia)
((follow promise see-Zerdia)   :name follow-promise)
((follow see-Zerdia attack-sportsman)    :name follow-see)
((success attack-sportsman)    :name success-attack)
((equals success-attack true)  :name successful-attack)
((cause has-feathers successful-attack)  :name cause-success-attack)
((realize Zerdia has-feathers) :name realize-Zerdia)
((follow successful-attack realize-Zerdia) :name follow-succ-attack)))
```

Dr. Phillip L. Ackerman
University of Minnesota
Department of Psychology
Minneapolis, MN 55455

Dr. Beth Adelson
Dept. of Computer Science
Tufts University
Medford, MA 02155

AFOSR,
Life Sciences Directorate
Bolling Air Force Base
Washington, DC 20332

Dr. Robert Ahlers
Code N711
Human Factors Laboratory
Naval Training Systems Center
Orlando, FL 32813

Dr. Ed Aiken
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. James Anderson
Brown University
Center for Neural Science
Providence, RI 02912

Dr. John R. Anderson
Department of Psychology
Carnegie-Mellon University
Pittsburgh, PA 15213

Dr. Steve Andriole
George Mason University
School of Information
Technology & Engineering
4400 University Drive
Fairfax, VA 22030

Technical Director, ARI
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Gary Aston-Jones
Department of Biology
New York University
1009 Main Bldg
Washington Square
New York, NY 10003

Dr. Patricia Baggett
University of Colorado
Department of Psychology
Box 345
Boulder, CO 80309

Dr. Eva L. Baker
UCLA Center for the Study
of Evaluation
145 Moore Hall
University of California
Los Angeles, CA 90024

Dr. Meryl S. Baker
Navy Personnel R&D Center
San Diego, CA 92152-6800

prof. dott. Bruno G. Bara
Unita di ricerca di
intelligenza artificiale
Universita di Milano
20122 Milano - via F. Sforza 23
ITALY

Dr. William M. Bart
University of Minnesota
Dept. of Educ. Psychology
330 Burton Hall
178 Pillsbury Dr., S.E.
Minneapolis, MN 55455

Leo Beltracchi
U.S. Nuclear Regulatory Comm.
Washington, D.C. 20555

Dr. Mark H. Bickhard
University of Texas
EDB 504 ED Psych
Austin, Texas 78712

Dr. Gautam Biswas
Department of Computer Science
University of South Carolina
Columbia, SC 29208

Dr. John Black
Teachers College, Columbia Univ.
525 West 121st Street
New York, NY 10027

Dr. R. Darrell Bock
University of Chicago
NORC
6030 South Ellis
Chicago, IL 60637

Dr. Sue Bogner
Army Research Institute
ATTN: PERI-SF
5001 Eisenhower Avenue
Alexandrai, VA 22333-5600

Dr. Jeff Bonar
Learning R&D Center
University of Pittsburgh
Pittsburgh, PA 15260

Dr. Gordon H. Bower
Department of Psychology
Stanford University
Stanford, CA 94306

Dr. Robert Breaux
Code N-095R
Naval Training Systems Center
Orlando, FL 32813

Dr. Ann Brown
Center for the Study of Reading
University of Illinois
51 Gerty Drive
Champaign, IL 61280

Dr. John S. Brown
XEROX Palo Alto Research
Center
3333 Coyote Road
Palo Alto, CA 94304

Dr. Bruce Buchanan
Computer Science Department
Stanford University
Stanford, CA 94305

Maj. Hugh Burns
AFHRL/IDE
Lowry AFB, CO 80230-5000

Dr. Patricia A. Butler
OERI
555 New Jersey Ave., NW
Washington, DC 20208

Dr. Joseph C. Campione
Center for the Study of Reading
University of Illinois
51 Gerty Drive
Champaign, IL 61820

Joanne Capper
Center for Research into Practice
1718 Connecticut Ave., N.W.
Washington, DC 20009

Dr. Jaime Carbonell
Carnegie-Mellon University
Department of Psychology
Pittsburgh, PA 15213

Dr. Susan Carey
Harvard Graduate School of Education
337 Gutman Library
Appian Way
Cambridge, MA 02138

Dr. Pat Carpenter
Carnegie-Mellon University
Department of Psychology
Pittsburgh, PA 15213

LCDR Robert Carter
Office of the Chief of Naval Operations
OP-01B
Pentagon
Washington, DC 20350-2000

Chair, Department of Psychology
College of Arts and Sciences
Catholic University of America
Washington, DC 20064

Dr. Fred Chang
Navy Personnel R&D Center
Code 51
San Diego, CA 92152-6800

Dr. Davida Charney
English Department
Penn State University
University Park, PA 16802

Dr. Paul R. Chatelier
OUSDRE
Pentagon
Washington, DC 20350-2000

Dr. Michelene Chi
Learning R & D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

Dr. Susan E. Chipman
1142CS
Personnel and Training Research Program
Office of Naval Research, Code 1142PT
Arlington, VA 22217-5000
(6 copies)

Dr. L. J. Chmura
Computer Science and Systems Branch
Naval Research Lab.
Washington, DC 20375-5000

Mr. Raymond E. Christal
AFHRL/MOE
Brooks AFB, TX 78235

Professor Chu Tien-Chen
Mathematics Department
National Taiwan University
Taipei, TAIWAN

Dr. Yee-Yeen Chu
Perceptronics, Inc.
21111 Erwin Street
Woodland Hills, CA 91367-3713

Dr. William Clancey
Stanford University
Knowledge Systems Laboratory
701 Welch Road, Bldg. C
Palo Alto, CA 94304

Dr. Charles Clifton
Tobin Hall
Department of Psychology
University of Massachusetts
Amherst, MA 01003

Dr. Allan M. Collins
Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, MA 02138

Dr. Stanley Collyer
Office of Naval Technology
Code 222
800 N. Quincy Street
Arlington, VA 22217-5000

Dr. William Crano
Department of Psychology
Texas A&M University
College Station, TX 77843

Brian Dallman
3400 TTW/TTGXS
Lowry AFB, CO 80230-5000

Dr. Laura Davis
NRL/NCARAI, Code 7510
4555 Overlook Ave., SW
Washington, DC 20375-5000

Dr. Natalie Dehn
Department of Computer and Information Science
University of Oregon
Eugene, OR 97403

Dr. Gerald F DeJong
Artificial Intelligence Group
Coordinated Science Laboratory
University of Illinois
Urbana, IL 61801

Goery Delacote
Directeur de L'informatique Scientifique et Technique
CNRS
15, Quai Anatole France
75700 Paris FRANCE

Dr. Sharon Derry
Florida State University
Department of Psychology
Tallahassee, FL 32306

Dr. Andrea di Sessa
University of California
School of Education
Tolman Hall
Berkeley, CA 94720

Dr. R. K. Dismukes
Associate Director for Life Sciences
AFOSR
Bolling AFB
Washington, DC 20332

Dr. Stephanie Doan
Code 6021
Naval Air Development Center
Warminster, PA 18974-5000

Dr. Emanuel Donchin
University of Illinois
Department of Psychology
Champaign, IL 61820

Defense Technical
Information Center
Cameron Station, Bldg 5
Alexandria, VA 22314
Attn: TC
(12 Copies)

Dr. Thomas M. Duffy
Communications Design Center
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. Richard Duran
University of California
Santa Barbara, CA 93106

Dr. John Ellis
Navy Personnel R&D Center
San Diego, CA 92252

Dr. Susan Embretson
University of Kansas
Psychology Department
426 Fraser
Lawrence, KS 66045

Dr. Randy Engle
Department of Psychology
University of South Carolina
Columbia, SC 29208

Dr. Susan Epstein
Hunter College
144 S. Mountain Avenue
Montclair, NJ 07042

ERIC Facility-Acquisitions
4833 Rugby Avenue
Bethesda, MD 20014

Dr. K. Anders Ericsson
University of Colorado
Department of Psychology
Boulder, CO 80309

Dr. Jean Claude Falmagne
Department of Psychology
New York University
New York, NY 10003

Dr. Beatrice J. Farr
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Pat Federico
Code 511
NPRDC
San Diego, CA 92152-6800

Dr. Paul Feltovich
Southern Illinois University
School of Medicine
Medical Education Department
P.O. Box 3926
Springfield, IL 62708

Mr. Wallace Feurzeig
Educational Technology
Bolt Beranek & Newman
10 Moulton St.
Cambridge, MA 02238

Dr. Gerhard Fischer
University of Colorado
Department of Computer Science
Boulder, CO 80309

J. D. Fletcher
9931 Corsica Street
Vienna VA 22180

Dr. Linda Flower
Carnegie-Mellon University
Department of English
Pittsburgh, PA 15213

Dr. Kenneth D. Forbus
University of Illinois
Department of Computer Science
1304 West Springfield Avenue
Urbana, IL 61801

Dr. Barbara A. Fox
University of Colorado
Department of Linguistics
Boulder, CO 80309

Dr. John R. Frederiksen
Bolt Beranek & Newman
50 Moulton Street
Cambridge, MA 02138

Dr. Norman Frederiksen
Educational Testing Service
Princeton, NJ 08541

Dr. Michael Friendly
Psychology Department
York University
Toronto Ontario
CANADA M3J 1P3

Julie A. Gadsden
Information Technology
Applications Division
Admiralty Research Establishment
Portsdown, Portsmouth PO6 4AA
UNITED KINGDOM

Dr. Michael Genesereth
Stanford University
Computer Science Department
Stanford, CA 94305

Dr. Dedre Gentner
University of Illinois
Department of Psychology
603 E. Daniel St.
Champaign, IL 61820

Chair, Department of
Psychology
George Mason University
Fairfax, VA 22030

Chair, Department of
Psychology
Georgetown University
Washington, DC 20057

Dr. Robert Glaser
Learning Research
& Development Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15260

Dr. Arthur M. Glenberg
University of Wisconsin
W. J. Brogden Psychology Bldg.
1202 W. Johnson Street
Madison, WI 53706

Dr. Sam Glucksberg
Department of Psychology
Princeton University
Princeton, NJ 08540

Dr. Susan Goldman
University of California
Santa Barbara, CA 93106

Dr. Sherrie Gott
AFHRL/MODJ
Brooks AFB, TX 78235

Dr. T. Govindaraj
Georgia Institute of Technology
School of Industrial & Systems
Engineering
Atlanta, GA 30332

Dr. Wayne Gray
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. James G. Greeno
University of California
Berkeley, CA 94720

Dr. Dik Gregory
Behavioral Sciences Division
Admiralty Research Establishment
Teddington, Middlesex
ENGLAND

Dr. Gerhard Grossing
Atominstitut
Schuttelstrasse 115
Vienna, AUSTRIA a-1020

Prof. Edward Haertel
School of Education
Stanford University
Stanford, CA 94305

Dr. Henry M. Halff
Halff Resources, Inc.
4918 33rd Road, North
Arlington, VA 22207

Dr. Ronald K. Hambleton
Prof. of Education & Psychology
University of Massachusetts
at Amherst
Hills House
Amherst, MA 01003

Stevan Harnad
Editor, The Behavioral and
Brain Sciences
20 Nassau Street, Suite 240
Princeton, NJ 08540

Dr. Wayne Harvey
SRI International
333 Ravenswood Ave.
Room B-S324
Menlo Park, CA 94025

Dr. Reid Hastie
Northwestern University
Department of Psychology
Evanston, IL 60201

Prof. John R. Hayes
Carnegie-Mellon University
Department of Psychology
Schenley Park
Pittsburgh, PA 15213

Dr. Barbara Hayes-Roth
Department of Computer Science
Stanford University
Stanford, CA 95305

Dr. Frederick Hayes-Roth
Teknowledge
525 University Ave.
Palo Alto, CA 94301

Dr. Shirley Brice Heath
School of Education
Stanford University
Stanford, CA 94305

Dr. Joan I. Heller
505 Haddon Road
Oakland, CA 94606

Dr. Jim Hollan
Intelligent Systems Group
Institute for
Cognitive Science (C-015)
UCSD
La Jolla, CA 92093

Dr. Melissa Holland
Army Research Institute for the
Behavioral and Social Sciences
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Keith Holyoak
University of Michigan
Human Performance Center
330 Packard Road
Ann Arbor, MI 48109

Ms. Julia S. Hough
Lawrence Erlbaum Associates
6012 Greene Street
Philadelphia, PA 19144

Dr. James Howard
Dept. of Psychology
Human Performance Laboratory
Catholic University of
America
Washington, DC 20064

Dr. Earl Hunt
Department of Psychology
University of Washington
Seattle, WA 98105

Dr. Ed Hutchins
Intelligent Systems Group
Institute for
Cognitive Science (C-015)
UCSD
La Jolla, CA 92093

Dr. Barbara Hutson
Virginia Tech
Graduate Center
2990 Telestar Ct.
Falls Church, VA 22042

Dr. Barbel Inhelder
University of Geneva
Geneva SWITZERLAND 12U-4

Dr. Dillon Inouye
WICAT Education Institute
Provo, UT 84057

Dr. Alice Isen
Department of Psychology
University of Maryland
Catonsville, MD 21228

Dr. Robert Jannarone
Department of Psychology
University of South Carolina
Columbia, SC 29208

Dr. Claude Janvier
Directeur, CIRADE
Universite' du Quebec a Montreal
Montreal, Quebec H3C 3P8
CANADA

Dr. Robin Jeffries
Hewlett-Packard Laboratories
P.O. Box 10490
Palo Alto, CA 94303-0971

Dr. Robert Jernigan
Decision Resource Systems
5595 Vantage Point Road
Columbia, MD 21044

Margaret Jerome
c/o Dr. Peter Chandler
83, The Drive
Hove
Sussex
UNITED KINGDOM

Chair, Department of
Psychology
The Johns Hopkins University
Baltimore, MD 21218

Dr. Douglas A. Jones
Thatcher Jones Assoc.
P.O. Box 6640
10 Trafalgar Court
Lawrenceville
NJ 08648

Dr. Marcel Just
Carnegie-Mellon University
Department of Psychology
Schenley Park
Pittsburgh, PA 15213

Dr. Daniel Kahneman
The University of British Columbia
Department of Psychology
#154-2053 Main Mall
Vancouver, British Columbia
CANADA V6T 1Y7

Dr. Ruth Kanfer
University of Minnesota
Department of Psychology
Elliott Hall
75 E. River Road
Minneapolis, MN 55455

Dr. Mary Grace Kantowski
University of Florida
Mathematics Education
359 Norman Hall
Gainesville, FL 32611

Dr. Milton S. Katz
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Frank Keil
Department of Psychology
Cornell University
Ithaca, NY 14850

Dr. Wendy Kellogg
IBM T. J. Watson Research Ctr.
P.O. Box 218
Yorktown Heights, NY 10598

Dr. Dennis Kibler
University of California
Department of Information
and Computer Science
Irvine, CA 92717

Dr. David Kieras
University of Michigan
Technical Communication
College of Engineering
1223 E. Engineering Building
Ann Arbor, MI 48109

Dr. Peter Kincaid
Training Analysis
& Evaluation Group
Department of the Navy
Orlando, FL 32813

Dr. Walter Kintsch
Department of Psychology
University of Colorado
Campus Box 345
Boulder, CO 80302

Dr. David Klahr
Carnegie-Mellon University
Department of Psychology
Schenley Park
Pittsburgh, PA 15213

Dr. Mazie Knerr
Program Manager
Training Research Division
HumRRO
1100 S. Washington
Alexandria, VA 22314

Dr. Janet L. Kolodner
Georgia Institute of Technology
School of Information
& Computer Science
Atlanta, GA 30332

Dr. Stephen Kosslyn
Harvard University
1236 William James Hall
33 Kirkland St.
Cambridge, MA 02138

Dr. Kenneth Kotovsky
Department of Psychology
Community College of
Allegheny County
800 Allegheny Avenue
Pittsburgh, PA 15233

Dr David H. Krantz
2 Washington Square Village
Apt. # 15J
New York, NY 10012

Dr. Benjamin Kuipers
University of Texas at Austin
Department of Computer Sciences
T.S. Painter Hall 3.28
Austin, Texas 78712

Dr. David R. Lambert
Naval Ocean Systems Center
Code 441T
271 Catalina Boulevard
San Diego, CA 92152-6800

Dr. Pat Langley
University of California
Department of Information
and Computer Science
Irvine, CA 92717

Dr. Marcy Lansman
University of North Carolina
The L. L. Thurstone Lab.
Davie Hall 013A
Chapel Hill, NC 27514

Dr. Jill Larkin
Carnegie-Mellon University
Department of Psychology
Pittsburgh, PA 15213

Dr. Jean Lave
School of Social Sciences
University of California
Irvine, CA 92717

Dr. Robert Lawler
Information Sciences, FRL
GTE Laboratories, Inc.
40 Sylvan Road
Waltham, MA 02254

Dr. Alan M. Lesgold
Learning R&D Center
University of Pittsburgh
Pittsburgh, PA 15260

Dr. Jim Levin
Dept. of Educational Psy.
210 Education Building
1310 South Sixth St.
Champaign, IL 61810-6990

Dr. John Levine
Learning R&D Center
University of Pittsburgh
Pittsburgh, PA 15260

Dr. Michael Levine
Educational Psychology
210 Education Bldg.
University of Illinois
Champaign, IL 61820

Dr. Clayton Lewis
University of Colorado
Department of Computer Science
Campus Box 430
Boulder, CO 80309

Matt Lewis
Department of Psychology
Carnegie-Mellon University
Pittsburgh, PA 15213

Library
Naval Training Systems Center
Orlando, FL 32813

Dr. Jane Malin
Mail Code SR 111
NASA Johnson Space Center
Houston, TX 77058

Dr. William L. Maloy
Chief of Naval Education
and Training
Naval Air Station
Pensacola, FL 32508

Dr. Sandra P. Marshall
Dept. of Psychology
San Diego State University
San Diego, CA 92182

Dr. Manton M. Matthews
Department of Computer Science
University of South Carolina
Columbia, SC 29208

Dr. Richard E. Mayer
Department of Psychology
University of California
Santa Barbara, CA 93106

Dr. Joe McLachlan
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. James McMichael
Assistant for MPT Research,
Development, and Studies
OP 01B7
Washington, DC 20370

Dr. Barbara Means
Human Resources
Research Organization
1100 South Washington
Alexandria, VA 22314

Dr. Douglas L. Medin
Department of Psychology
University of Illinois
603 E. Daniel Street
Champaign, IL 61820

Dr. George A. Miller
Department of Psychology
Green Hall
Princeton University
Princeton, NJ 08540

Dr. William Montague
NPRDC Code 13
San Diego, CA 92152-6800

Dr. Allen Munro
Behavioral Technology
Laboratories - USC
1845 S. Elena Ave., 4th Floor
Redondo Beach, CA 90277

Chair, Department of
Computer Science
U.S. Naval Academy
Annapolis, MD 21402

Dr. Allen Newell
Department of Psychology
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. Richard E. Nisbett
University of Michigan
Institute for Social Research
Room 5261
Ann Arbor, MI 48109

Dr. Mary Jo Nissen
University of Minnesota
N218 Elliott Hall
Minneapolis, MN 55455

Director, Training Laboratory,
NPRDC (Code 05)
San Diego, CA 92152-6800

Director, Manpower and Personnel
Laboratory,
NPRDC (Code 06)
San Diego, CA 92152-6800

Director, Human Factors
& Organizational Systems Lab,
NPRDC (Code 07)
San Diego, CA 92152-6800

Fleet Support Office,
NPRDC (Code 301)
San Diego, CA 92152-6800

Library, NPRDC
Code P201L
San Diego, CA 92152-6800

Dr. Harold F. O'Neil, Jr.
School of Education - WPH 801
Department of Educational
Psychology & Technology
University of Southern California
Los Angeles, CA 90089-0031

Dr. Michael Oberlin
Naval Training Systems Center
Code 711
Orlando, FL 32813-7100

Dr. Stellan Ohlsson
Learning R & D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

Office of Naval Research,
Code 1133
800 N. Quincy Street
Arlington, VA 22217-5000

Office of Naval Research,
Code 1142
800 N. Quincy St.
Arlington, VA 22217-5000

Psychologist
Office of Naval Research
Branch Office, London
Box 39
FPO New York, NY 09510

Special Assistant for Marine
Corps Matters,
ONR Code 00MC
800 N. Quincy St.
Arlington, VA 22217-5000

Psychologist
Office of Naval Research
Liaison Office, Far East
APO San Francisco, CA 96503

Dr. Judith Orasanu
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Prof. Seymour Papert
20C-109
Massachusetts Institute
of Technology
Cambridge, MA 02139

Dr. James Paulson
Department of Psychology
Portland State University
P.O. Box 751
Portland, OR 97207

Dr. Roy Pea
Bank Street College of
Education
610 W 112th Street
New York, NY 10025

Dr. Douglas Pearse
DCIEM
Box 2000
Downsview, Ontario
CANADA

Dr. James W. Pellegrino
University of California,
Santa Barbara
Department of Psychology
Santa Barbara, CA 93106

Dr. Virginia E. Pendergrass
Code 711
Naval Training Systems Center
Orlando, FL 32813-7100

Military Assistant for Training and
Personnel Technology,
OUSD (R & E)
Room 3D129, The Pentagon
Washington, DC 20301-3080

Dr. David N. Perkins
Educational Technology Center
337 Gutman Library
Appian Way
Cambridge, MA 02138

Dr. Nancy Perry
Chief of Naval Education
and Training, Code 00A2A
Naval Station Pensacola
Pensacola, FL 32508

Department of Computer Science,
Naval Postgraduate School
Monterey, CA 93940

Dr. Steven Pinker
Department of Psychology
E10-018
M.I.T.
Cambridge, MA 02139

Dr. Tjeerd Plomp
Twente University of Technology
Department of Education
P.O. Box 217
7500 AE ENSCHEDE
THE NETHERLANDS

Dr. Martha Polson
Department of Psychology
Campus Box 346
University of Colorado
Boulder, CO 80309

Dr. Peter Polson
University of Colorado
Department of Psychology
Boulder, CO 80309

Dr. Steven E. Poltrock
MCC
9430 Research Blvd.
Echelon Bldg #1
Austin, TX 78759-6509

Dr. Harry E. Pople
University of Pittsburgh
Decision Systems Laboratory
1360 Scaife Hall
Pittsburgh, PA 15261

Dr. Mary C. Potter
Department of Psychology
MIT (E-10-032)
Cambridge, MA 02139

Dr. Joseph Psotka
ATTN: PERI-1C
Army Research Institute
5001 Eisenhower Ave.
Alexandria, VA 22333

Dr. Lynne Reder
Department of Psychology
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. James A. Reggia
University of Maryland
School of Medicine
Department of Neurology
22 South Greene Street
Baltimore, MD 21201

Dr. Fred Reif
Physics Department
University of California
Berkeley, CA 94720

Dr. Lauren Resnick
Learning R & D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

Dr. Gil Ricard
Mail Stop C04-14
Grumman Aerospace Corp.
Bethpage, NY 11714

Mark Richer
1041 Lake Street
San Francisco, CA 94118

Dr. Mary S. Riley
Program in Cognitive Science
Center for Human Information Processing
University of California
La Jolla, CA 92093

Dr. Linda G. Roberts
Science, Education, and Transportation Program
Office of Technology Assessment
Congress of the United States
Washington, DC 20510

Dr. William B. Rouse
Search Technology, Inc.
25-b Technology Park/Atlanta
Norcross, GA 30092

Dr. David Rumelhart
Center for Human Information Processing
Univ. of California
La Jolla, CA 92093

Dr. Roger Schank
Yale University
Computer Science Department
P.O. Box 2158
New Haven, CT 06520

Dr. Walter Schneider
Learning R&D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15260

Dr. Alan H. Schoenfeld
University of California
Department of Education
Berkeley, CA 94720

Dr. Janet Schofield
Learning R&D Center
University of Pittsburgh
Pittsburgh, PA 15260

Karen A. Schriver
Department of English
Carnegie-Mellon University
Pittsburgh, PA 15213

Dr. Judah L. Schwartz
MIT
20C-120
Cambridge, MA 02139

Dr. Marc Sebrechts
Department of Psychology
Wesleyan University
Middletown, CT 06475

Dr Judith Segal
OERI
555 New Jersey Ave., NW
Washington, DC 20208

Dr Sylvia A. S. Shafto
Department of Computer Science
Towson State University
Towson, MD 21204

Dr. Ben Shneiderman
Dept. of Computer Science
University of Maryland
College Park, MD 20742

Dr. Lee Shulman
Stanford University
1040 Cathcart Way
Stanford, CA 94305

Dr. Robert S. Siegler
Carnegie-Mellon University
Department of Psychology
Schenley Park
Pittsburgh, PA 15213

Dr. Derek Sleeman
Stanford University
School of Education
Stanford, CA 94305

Dr. Edward E. Smith
Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, MA 02138

Dr. Richard E. Snow
Department of Psychology
Stanford University
Stanford, CA 94306

Dr. Elliot Soloway
Yale University
Computer Science Department
P.O. Box 2158
New Haven, CT 06520

Dr. Richard Sorensen
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. Kathryn T. Spoehr
Brown University
Department of Psychology
Providence, RI 02912

Dr Robert Sternberg
Department of Psychology
Yale University
Box 11A, Yale Station
New Haven, CT 06520

Dr. Albert Stevens
Bolt Beranek & Newman, Inc.
10 Moulton St.
Cambridge, MA 02238

Dr. Thomas Sticht
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. John Tangney
AFOSR/NL
Bolling AFB, DC 20332

Dr. Kikumi Tatsuoka
CERL
252 Engineering Research
Laboratory
Urbana, IL 61801

Dr. Perry W. Thorndyke
FMC Corporation
Central Engineering Labs
1185 Coleman Avenue, Box 580
Santa Clara, CA 95052

Dr. Douglas Towne
Behavioral Technology Labs
1845 S. Elena Ave.
Redondo Beach, CA 90277

Chair, Department of
Computer Science
Towson State University
Towson, MD 21204

Chair, Department of
Psychology
Towson State University
Towson, MD 21204

Dr. Kurt Van Lehn
Department of Psychology
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213

Dr. Beth Warren
Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, MA 02138

Dr. Donald Weitzman
MITRE
1820 Dolley Madison Blvd.
MacLean, VA 22102

Dr. Keith T. Wescourt
FMC Corporation
Central Engineering Labs
1185 Coleman Ave., Box 580
Santa Clara, CA 95052

Dr. Douglas Wetzel
Code 12
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. Barbara White
Bolt Beranek & Newman, Inc.
10 Moulton Street
Cambridge, MA 02238

Dr. Christopher Wickens
Department of Psychology
University of Illinois
Champaign, IL 61820

Dr. Heather Wild
Naval Air Development Center
Code 6021
Warminster, PA 18974-5000

Dr. Michael Williams
IntelliCorp
1975 El Camino Real West
Mountain View, CA 94040-2216

Dr. Robert A. Wisher
U.S. Army Institute for the
Behavioral and Social Sciences
5001 Eisenhower Avenue
Alexandria, VA 22333

Mr. John H. Wolfe
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. Wallace Wulfeck, III
Navy Personnel R&D Center
San Diego, CA 92152-6800

Dr. Joe Yasatuke
AFHRL/LRT
Lowry AFB, CO 80230

Dr. Masoud Yazdani
Dept. of Computer Science
University of Exeter
Exeter EX4 4QL
Devon, ENGLAND

Mr. Carl York
System Development Foundation
181 Lytton Avenue
Suite 210
Palo Alto, CA 94301

Dr. Joseph L. Young
Memory & Cognitive
Processes
National Science Foundation
Washington, DC 20550